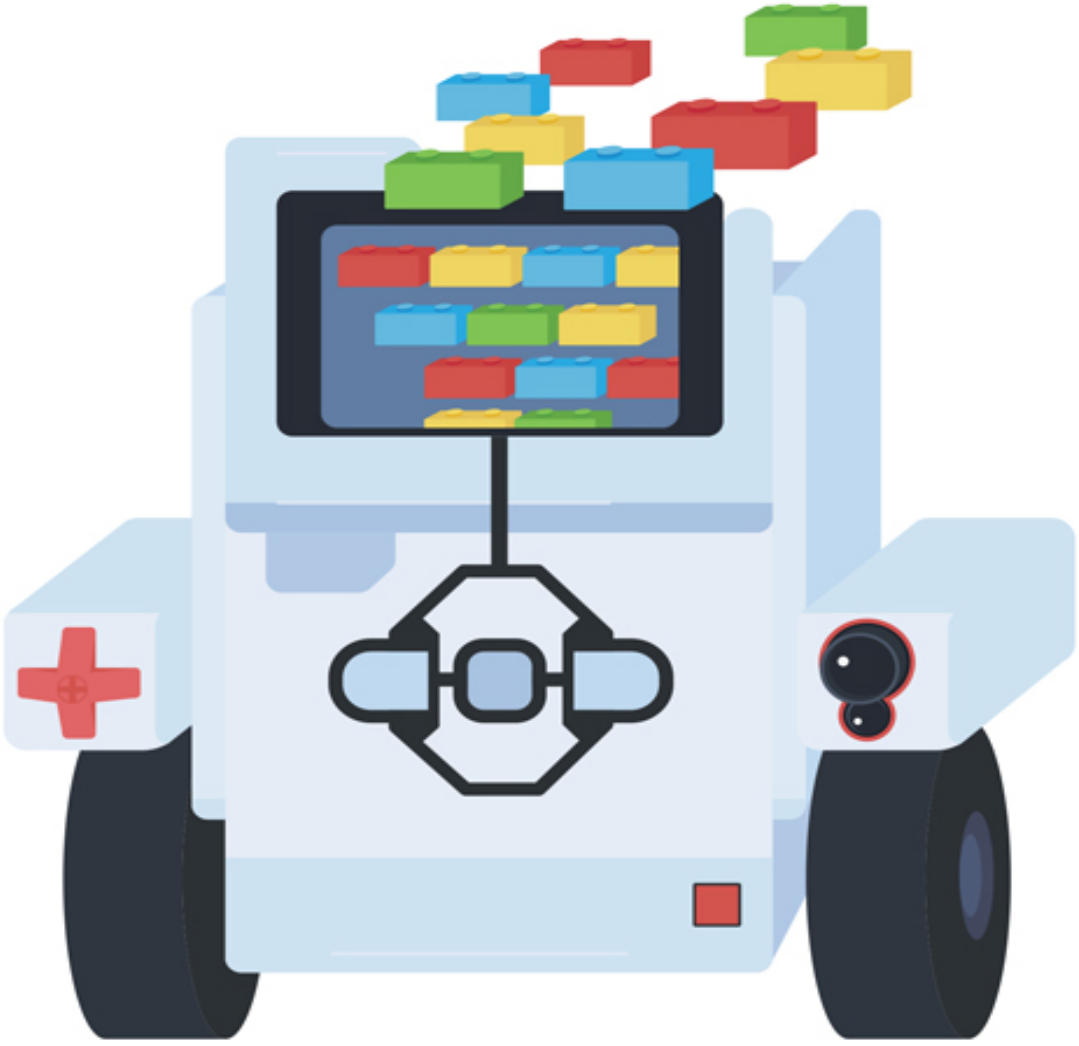
# Introdução ao kit robótico

# LEGO® EV3

## Programe seus robôs com linguagem de blocos



Casa do Código

LUIZ ROQUE
VITOR GONÇALVES

# Sumário


- ISBN
- Prefácio
- Sobre os autores
- Agradecimentos
- Dedicatória
- O HARDWARE
  - 1 O kit LEGO® MINDSTORMS® EV3
- O SOFTWARE
  - 2 Instalando o software LEGO® MINDSTORMS® EV3 Home Edition
  - 3 Instalando o software LEGO® MINDSTORMS® Education EV3
  - 4 O software LEGO® MINDSTORMS® EV3
  - 5 Utilizando o software EV3 no NXT 2.0
  - 6 Tools
  - 7 Conclusão
- A LINGUAGEM EV3-G
  - 8 Conhecendo o ambiente de programação
  - 9 Blocos de programação
  - 10 Utilização e configuração dos blocos
  - 11 Criando e executando um programa no Brick EV3
  - 12 Conclusão
- PRIMEIROS PROGRAMAS
  - 13 Medidor de distâncias
  - 14 Gerador de números aleatórios
  - 15 Identificador de cores
  - 16 Medidor de luz refletida
  - 17 Detector de presença
  - 18 Conclusão
- EXPERIÊNCIAS
  - 19 Seguidor de Linha
  - 20 Sumô

    - 21 Explorer
    - 22 Conclusão
- RECURSOS AVANÇADOS
    - 23 Boas práticas de programação no EV3
    - 24 Organização das tarefas
    - 25 Divisão de tarefas
    - 26 Calibrando o Colour Sensor
    - 27 NXT – Utilizando o sensor de luminosidade
    - 28 Aprimorando - Seguidor de Linha com bloco Wait
    - 29 Aprimorando - Sumô com sensor ultrassônico
    - 30 Aprimorando - Explorer com o Gyro Sensor
    - 31 Conclusão
- E AGORA?
    - 32 Sites para seguir seus estudos
    - 33 Competições
    - 34 Explorando
    - 35 Novas linguagens e ambientes de programação
    - 36 Nota final
    - 37 Links úteis

## ISBN

Impresso e PDF: 978-85-94188-84-7

EPUB: 978-85-94188-85-4

MOBI: 978-85-94188-86-1


Caso você deseje submeter alguma errata ou sugestão, acesse http://erratas.casadocodigo.com.br.

# Prefácio

Os kits robóticos da LEGO® vêm alcançando um espaço cada vez maior na educação e aprendizagem. Dentro os fatores que mais contribuem para sua popularidade estão sua versatilidade e as inúmeras possibilidades que eles abrem. Os kits robóticos da LEGO® podem ser usados por professores, alunos e pais; pode ser utilizado como ferramenta educacional em salas de aula ou até mesmo em casa. Além disso, os kits são usados por diversos competidores em olimpíadas de robótica.

Muito conteúdo se encontra sobre o kit robótico LEGO® MINDSTORMS® EV3 em sites e livros, por exemplo. Porém, a grande maioria desse conteúdo está em inglês. Observando este cenário, este livro visa fornecer uma ferramenta de aprendizagem em nosso idioma. E, principalmente, ser uma porta de entrada para as pessoas que estão iniciando e anseiam por aprender mais acerca do kit LEGO® MINDSTORMS® EV3.

O próprio software EV3 (LEGO® MINDSTORMS® Education EV3) oferece uma vasta gama de recursos e informações para que você dê seus primeiros passos e aprenda mais sobre programação. O objetivo deste livro é servir como uma introdução, ele pode ser utilizado até mesmo por aqueles que nunca tiveram contato com o kit ou com programação antes. Vamos fornecer uma porta de entrada para o leitor e avançaremos ao longo do livro.

Como a obra foi pensando para atender, até mesmo, os leitores que estão tendo contato com o kit pela primeira ou, ainda, para aqueles que não decidiram qual versão comprar; antes de iniciar seu assunto principal, que é a programação, o livro oferece uma introdução sobre os dois kits disponíveis no mercado: o Home Edition e a versão Education. Portanto, se você ainda não se decidiu qual versão do kit comprar, este livro pode lhe oferecer um bom esclarecimento.

Durante a produção deste livro, pensou-se também nos usuários da versão anterior do kit, os que possuem o kit NXT 2.0. Por isso, se você possui esse kit saiba que é possível programá-lo utilizando o software LEGO® MINDSTORMS® Education EV3. Para isso, basta seguir alguns passos, que são explicados no livro. Apesar de ter saído de linha, o kit NXT 2.0 ainda possui uma grande quantidade de usuários, devido ao seu baixo custo (justamente pelo fato de ter saído de linha). Então, se você possui o kit NXT 2.0 e quer aprender a programar (de um modo geral), ou quer aprender a linguagem EV3-L e desfrutar de todos os recursos, este é um bom livro para você.

Este livro aborda sobretudo a programação, seu objetivo é ensinar a linguagem EV3-G (nome da linguagem de blocos) de maneira didática e prática. Você encontrará desde projetos que requerem apenas o Brick e alguns sensores, até robôs mais sofisticados, como o seguidor de linha, sumô e o explorer.

O foco do livro não é a construção dos protótipos, mas sim ensinar programação. Por isso, ao decorrer dos capítulos, sinta-se à vontade para realizar as modificações que achar necessárias e que trarão melhoria ao seu robô. Um dos fatores mais incríveis do kit é justamente isso, a facilidade que ele oferece e a possibilidade de criar robôs incríveis.

Ao final do livro, você também encontrará dicas, sugestões e materiais de apoio para que você possa continuar seus estudos e progredir cada vez mais. Há um capítulo que destaca algumas das ferramentas alternativas que podem ser utilizadas para programar o kit, bem como suas linguagens de programação. Ao final da leitura, você verá que as possibilidades com o kit LEGO® MINDSTORMS® são imensas e que, o kit é uma ótima forma de aprender a programar.

# Sobre os autores

## Vitor Emanuel Gonçalves

Vitor é programador há 6 anos, tendo despertado grande interesse por estudar a partir de seus primeiros anos de vida. Suas características de autodidata e pesquisador o levaram a estudar no Instituto Federal Fluminense (IFF) de Macaé, onde atualmente faz o curso técnico de automação industrial. Ingressando no Laboratório de Automação do IFF em junho de 2017, Vitor ampliou ainda mais sua afinidade em pesquisa, tendo desenvolvido por iniciativa própria apostilas em diversas áreas do conhecimento. Bastante querido pelos alunos frequentadores do laboratório, Vitor se destaca por sua educação e boa vontade, sempre disposto a ajudar alunos em diferentes níveis de estudos.

Ao tomar conhecimento dos projetos de robótica de seu orientador, desenvolvidos na plataforma LEGO® MINDSTORMS® NXT 2.0, Vitor percebeu que havia lacuna nas publicações nacionais do novo conjunto educacional robótico da empresa dinamarquesa. Dessa forma, ele resolveu escrever este livro sobre programação em linguagem de blocos do kit EV3, que se destaca por redação objetiva, clara e didática, tendo concluído em poucos meses de árduo trabalho sua publicação, antes de completar 18 anos.

## Luiz Alberto Oliveira Lima Roque

Luiz é professor dos cursos de Engenharia de Controle e Técnico em Automação Industrial, atuando desde 2008 no Campus Macaé, unidade integrante do Instituto Federal Fluminense de Educação, Ciência e Tecnologia. É mestre em Engenharia de Computação e Engenheiro Eletricista, ambos os cursos concluídos pela Universidade do Estado do Rio de Janeiro, onde também exerceu o ofício de programador de computadores por 16 anos. Possui especializações em Robótica, Mecatrônica, Automação, Eletrotécnica, Geoprocessamento, Engenharia de Segurança,

Engenharia Ambiental, Educação Profissional e Tecnológica, Educação Especial com Ênfase em Surdez, Gestão de Educação a Distância, Física, Novas Tecnologias no Ensino da Matemática, Gestão de Cidades e Educação Física com Ênfase em Taekwondo - 3° Dan. Já exerceu funções de coordenador de pesquisa, do curso técnico de automação e diretor substituto de inovação, no IFF – Macaé. Lecionou disciplinas como Eletrônica, Laboratório de Eletrônica, Robótica, Informática Industrial, Sistemas Supervisórios, Comunicação de Dados, Protocolos de Redes Industriais, Controladores Lógicos Programáveis, Modelagem Matemática com MatLab, Inteligência Artificial, Sistemas Digitais, Trabalho de Conclusão de Curso, Algoritmos e Técnicas de Programação e Matemática Discreta.

O docente é autor do livro *Automa o de Processos com Linguagem Ladder e Sistemas Supervis rios*, publicado pela editora Livros Técnicos e Científicos (LTC), no ano de 2014, obra que se encontra em sua segunda reimpressão. Em 2017 foi lançada sua segunda obra, cujo título é *Uma Arquitetura de Monitoramento Ambiental*, pela editora Itacaiúnas. Luiz Roque também se dedica à redação de artigos científicos, e teve a honra de publicar dois trabalhos na Sociedade Internacional de Sensoriamento Remoto e Geografia, pela Instituto Internacional de Engenharia Elétrica e Eletrônica (IEEE), em 2004. Um dos artigos apresentados no evento propunha um analisador léxico e sintático para a Linguagem de Marcação Geográfica (GML).

# Agradecimentos

Muitas pessoas contribuíram para a elaboração deste livro, por isso, o mínimo que posso fazer é agradecer-lhes. Agradeço pelo apoio, pelo carinho e, por sempre me incentivarem e servirem de fonte de inspiração.

Na verdade, isto é mais que um agradecimento. Isto é uma expressão de carinho e gratidão por todos vocês, pois sem vocês este livro não seria o que ele se tornou. Por isso, registro, aqui, meu carinho e minha singela homenagem a todos vocês.

Ao professor Luiz Alberto Roque, que é coautor, amigo e excelente mestre, com quem eu sempre aprendo cada vez mais. Seu apoio e boa vontade ultrapassam o âmbito escolar, e são fundamentais para mim. Agradeço pela confiança e por todo o apoio exercido até aqui.

Agradeço também ao Pedro Herdy, pela ajuda com as fotos, pelo incentivo e, pela pronta disponibilidade a me ajudar com os problemas que surgiram durante essa jornada e, por ser um ótimo colega de trabalho.

A Débora, Heloísa, Marcelo, Sandra, Lia e Max, por todo o apoio prestado até aqui, nos momentos mais difíceis e, por sempre acreditarem em mim. Vocês foram fundamentais para que eu pudesse prosseguir.

Aos professores, Gladstone, Denaldo, João Álvaro, Luiz Viégas e Rafael, em primeiro lugar, pelas excelentes pessoas que são e, em segundo, mas não menos importante, por serem excelentes professores e por todo o apoio prestado durante minha jornada no IFF Macaé.

Em suma, agradeço a toda equipe do IFF, por fazerem deste um lugar incrível, de conhecimento e aprendizagem. Obrigado a cada de um vocês, docentes e discentes.

Agraço também à Vivian, editora deste livro, sem ela o livro não chegaria nem perto deste resultado final obtido. Com sua experiência, seus comentários e dicas, o livro foi se consolidando e ganhando cada vez mais qualidade.

Quero que todos saibam que vocês foram fundamentais em minha vida. Sou muito feliz e grato por ter a honra e o prazer de conviver com vocês, muito obrigado!

- Vitor Emanuel Gonçalves

# Dedicatória

Dedico este livro à minha mãe, que sempre apoiou meus planos, e a Deus pela graça de conseguir a publicação desta obra.

# O HARDWARE

Antes de começar a programar, é fundamental que o usuário conheça e entenda a parte física do seu kit. É imprenscindível o conhecimento sobre o funcionamento dos seus componentes, bem como as funções que podem ser desempenhadas pelo Brick (CPU) e os modos de funcionamento dos sensores.

O primeiro passo para o aprendizado de programação, portanto, será conhecer um pouco sobre o funcionamento de cada sensor e sobre o Brick. Por isso, esta parte do livro contém um capítulo em que é abordado cada componente (sensores, cabos etc.) e sua função.

## Capítulo 1
## O kit LEGO® MINDSTORMS® EV3

## 1.1 O EV3

O conjunto robótico EV3 é o mais recente kit educacional da LEGO®, constituindo-se sucessor da geração anterior, conhecida por NXT. O nome EV3 se refere à **ev**olução e o número 3 indica que ele é o terceiro da linha.

O primeiro conjunto educacional para ensino de robótica desenvolvido pela LEGO® foi o RCX, formado por peças encaixáveis, engrenagens, correias, sensores e atuadores, agregados ao módulo Brick, que é composto por unidade central de processamento, memórias e interfaces de entrada e saída. As demais gerações de kits robóticos da LEGO® seguiram o paradigma iniciado pelo protótipo RCX, sendo o modelo EV3 anunciado em janeiro de 2013, e comercializado em dezembro do mesmo ano.

Figura 1.1: O kit LEGO® EV3

O EV3 apresentou diversas mudanças, ampliando ainda mais as possibilidades na área de robótica educacional. Enquanto o NXT possui um processador ARM7, a nova linha EV3 apresenta o processador ARM9, possibilitando a execução de um sistema operacional Linux. O módulo ainda conta com um slot para cartões SD de até 32 GB, juntamente com um conector USB. Todas essas novidades contribuem para a criação de protótipos cada vez mais sofisticados, proporcionando o desenvolvimento de projetos que seriam quase impossíveis de serem realizados no módulo NXT.

O novo módulo é mais robusto, com visual ousado e moderno, quando comparado em relação às gerações anteriores. O Brick Status Light é um novo incremento presente no EV3, com a função de exibir o status do módulo por meio de cores, por meio do LED ao

redor dos botões. Outro recurso ainda melhor consistiu na adição de uma quarta porta de saída, permitindo o uso de mais um motor.

Além disso, no kit EV3 os sensores de cor atuam em duas funções, que permitem identificar cores (quando configurados no modo de operação *sensor de cor*) ou medir a intensidade luminosa ambiente (quando ajustados no modo de operação *sensor de lu* ), pela luz refletida pelo LED vermelho.

O EV3 é comercializado em duas versões principais: LEGO® MINDSTORMS® EV3 Home Edition e a versão LEGO® MINDSTORMS® Education EV3 Core Set. (Ao decorrer do livro será utilizada a abreviação "Education".)

A versão EV3 Home Edition é mais voltada para um aprendizado de forma mais divertida e lúdica. Esta versão conta com menos sensores que a Education. Porém, este kit apresenta a possibilidade de criar cinco modelos de robôs disponibilizados pela LEGO®, isto é encontrado na área inicial do seu software, que apresenta as instruções para cada um. Os modelos são: EV3RSTORM, GRIPP3R, R3PTAR, SPIK3R e TRACK3R. A versão conta com 585 peças encaixáveis para montagem, um módulo EV3, dois motores grandes, um motor médio, um sensor de toque, um sensor de cor, um sensor infravermelho acompanhado de controle, cabos conectores e um cabo USB, que é utilizado para fazer a conexão com o computador. O software da versão Home é o LEGO® MINDSTORMS® EV3 Home Edition.

Já a versão Education EV3 Core Set é mais indicada para a robótica educacional, com um uso em sala de aula, ou para competições. Dessa forma, esta versão é a melhor opção para usuários com um pouco mais de experiência. Ela apresenta mais possibilidades de criações próprias de robôs, pois conta com mais sensores. O kit tem 547 peças de montagem, um módulo EV3, bateria recarregável, dois motores grandes, um motor médio, dois sensores de toque, um sensor giroscópio, um sensor ultrassônico, um sensor de cor, conectores e um cabo USB.

Sensores e motores do conjunto robótico NXT funcionam em modelos EV3. Entretanto, a recíproca não é verdadeira, uma vez que sensores projetados para o LEGO® EV3 não funcionam no NXT, embora os motores da geração atual sejam reconhecidos pelos módulos NXT, podendo ser utilizados neles. Enfim, com o kit EV3 o leque de possibilidades e aprendizagem tornou-se ainda mais vasto.

## 1.2 Componentes do kit

Conhecer os componentes do kit MINDSTORMS® EV3 é fundamental para que se possa programá-los. Nesta seção, daremos uma pincelada a respeito deles.

### O Brick

O módulo Brick EV3 compreende a CPU (unidade central de processamento) do kit, juntamente com interfaces de entrada e saída. Por meio desse módulo é feita a conexão entre hardware e software. O Brick possui o display, 6 botões, 4 portas de entrada e 4 portas de saída, mais uma porta de conexão USB.

Nas portas de entrada (*Input Ports*) são inseridos os sensores, enquanto nas portas de saída (*Output Ports*) são conectados os servo-motores, ou simplesmente motores (a descrição detalhada dos motores será dada em breve). A porta USB realiza a conexão entre o Brick e o computador, permitindo a compilação dos programas no ambiente de desenvolvimento integrado, e posterior carregamento à CPU do EV3. Os botões servem para interagir com a interface do Brick, mas também podem ser usados como sensores.

Figura 1.2: O BRICK

O Brick apresenta:

### Brick Sound (Alto falante do Brick)

Usa-se o Brick Sound para reproduzir sons carregados na memória do Brick. Trata-se de um alto falante acoplado internamente no Brick.

### Brick Status Light (Status de Luz do Brick)

Ao se utilizar o Brick, percebe-se que luzes de diferentes cores são acesas ao redor dos botões. Essas luzes indicam o status do Brick, informando em que condição ele se encontra. As cores têm os seguintes significados:

- Vermelho – Quando o Brick se inicia, está em update ou desligando.
- Vermelho pulsante – Quando o Brick está ocupado.
- Laranja – Estado de alerta, pronto.
- Laranja pulsante – Alerta de execução.
- Verde – Pronto.
- Verde pulsante – O programa está sendo executado.

### Brick Display (Display do Brick)

O display é a tela do Brick, onde é possível exibir textos e imagens. Isso é feito por meio de um bloco de programação.

### Cabos

Há dois tipos de cabos no kit, o cabo responsável pela comunicação entre o PC e o Brick, e os cabos responsáveis pela comunicação entre o Brick e os sensores e motores.

### Cabo USB

Este cabo realiza a comunicação entre o PC e o Brick. O modelo EV3 reformulou o cabo, portanto este cabo é diferente do cabo do kit NXT. A porta PC está localizada próxima à porta D.

Figura 1.3: Cabo USB

## Cabos conectores

Utilizam-se estes cabos para realizar a comunicação dos sensores e também dos motores com o Brick. O kit apresenta distintos tamanhos de cabos, que podem ser curtos, médios ou longos.

Figura 1.4: Cabos conectores

## Alimentação

O kit LEGO® MINDSTORMS® Education EV3 conta com uma bateria de lítio, que pode ser acoplada no lugar das pilhas. O modelo Home Basic não conta com bateria. Porém, nada impede que uma bateria seja adquirida separadamente. Nenhuma das versões conta com carregadores, mas você pode adquiri-los separadamente.

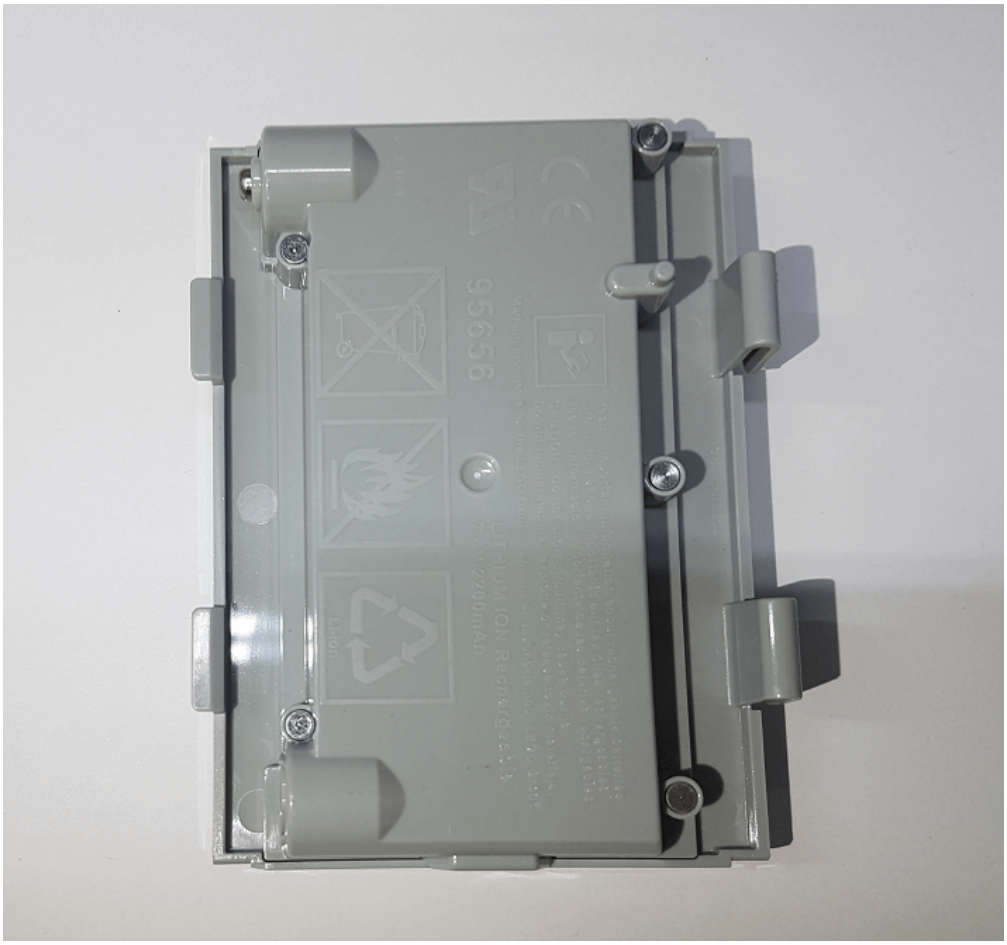



Figura 1.5: Bateria

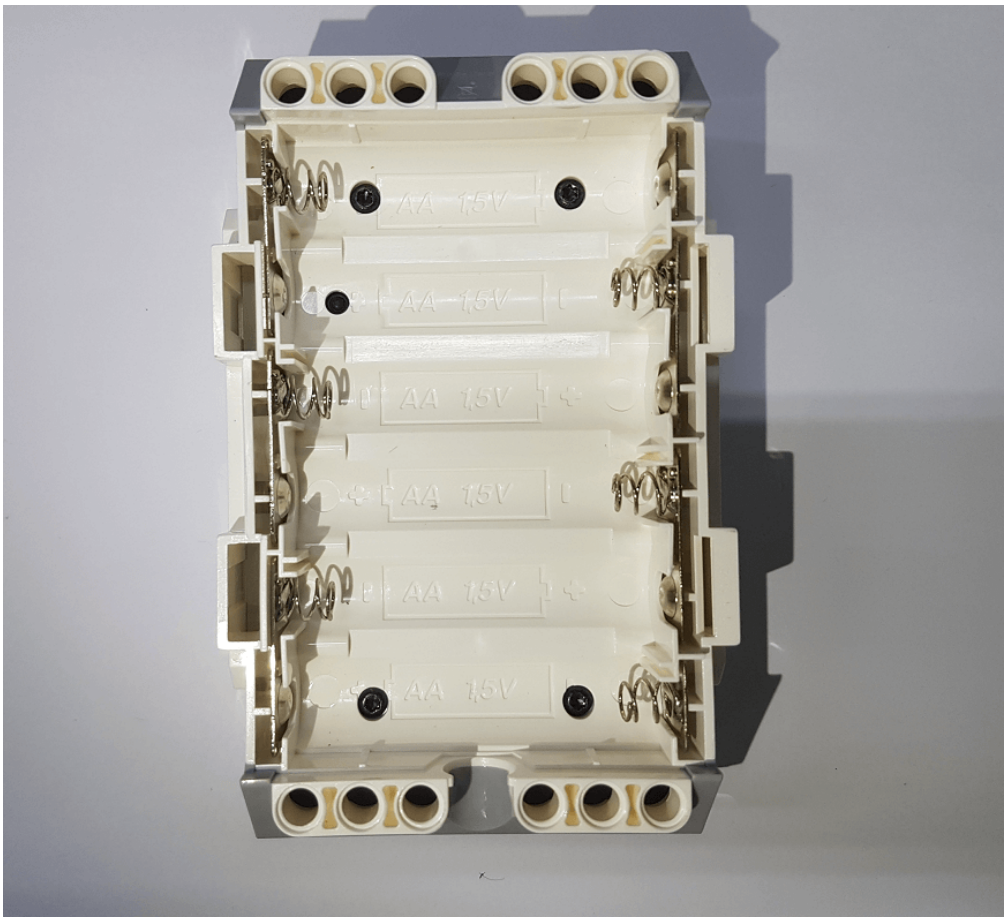



Figura 1.6: Parte inferior do Brick -Entrada para pilhas ou bateria

## Manual

O modelo Education vem acompanhado de manual de montagem para um robô, contendo as devidas instruções. O robô apresentando no guia é bem simples, porém constitui uma ótima oportunidade de o usuário se iniciar na fascinante aventura de montar e programar protótipos robóticos. A versão Home Basic do EV3 também conta com manual, que contém instruções para um dos robôs utilizados em missões (*Robot Missions*).

Figura 1.7: Manual Home Edition

Figura 1.8: Manual Education

## Motores

## Large Motor (Motor grande)

Trata-se de um motor potente, equipado com um sensor de rotação, ou *Rotation Sensor*. Este motor é mais lento, pois executa menos rotações por minuto. Porém, é mais forte, devido ao maior torque apresentado.

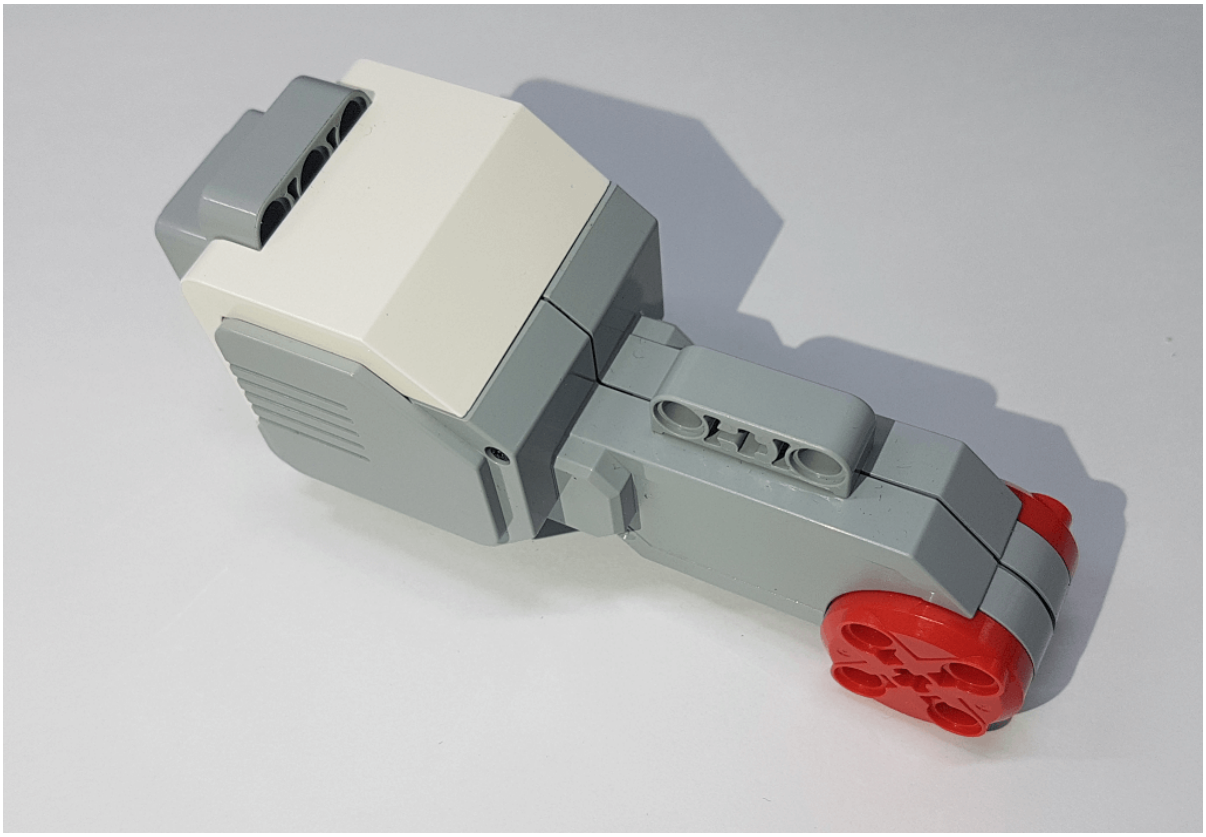


Figura 1.9: Large Motor (Motor grande)

## Medium Motor (Motor médio)

É um motor mais leve e compacto, caracterizado por resposta rápida. Ele executa mais rotações por minuto e tem um torque menor. Também é equipado com um sensor de rotação.

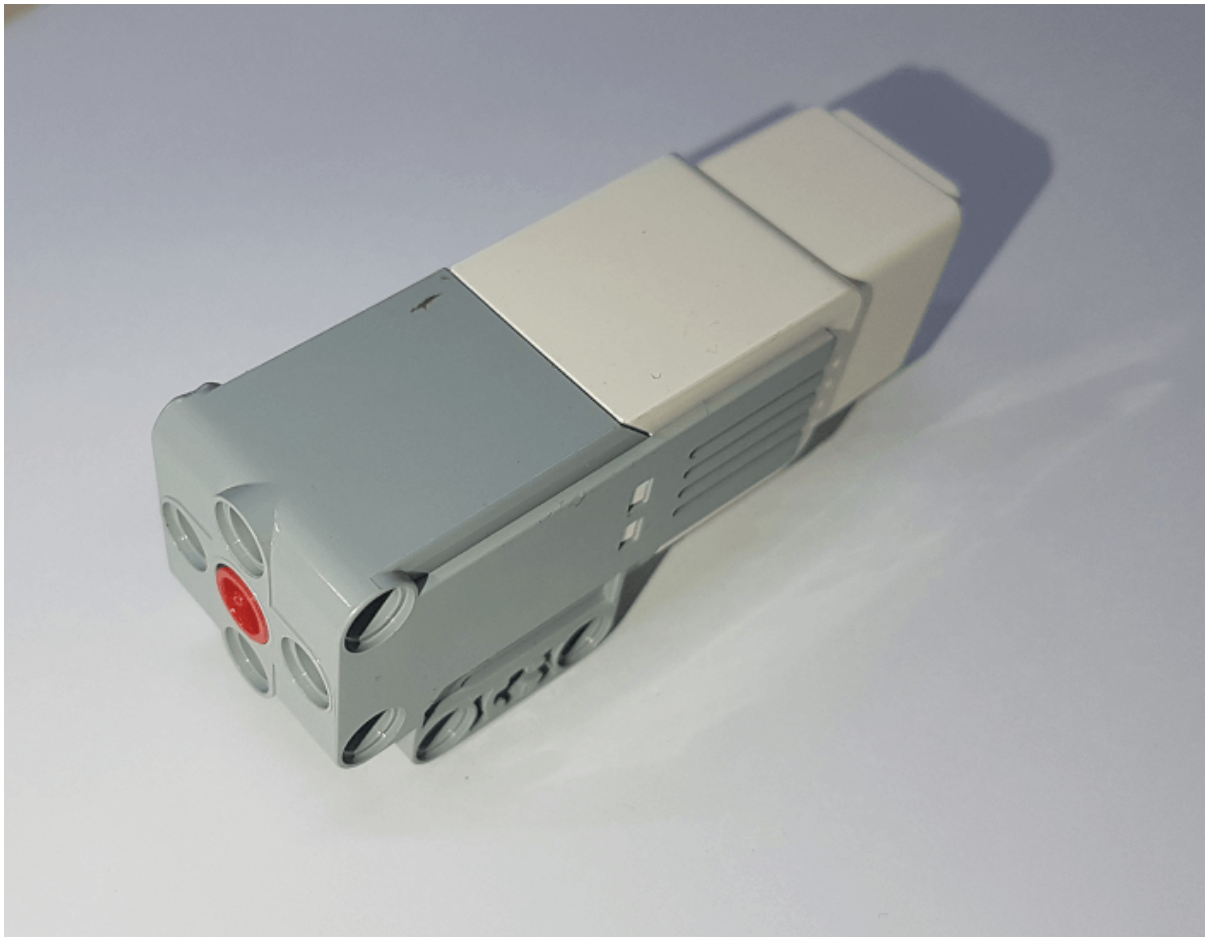


Figura 1.10: Medium Motor (Motor médio)

## Sensores

## Touch Sensor (Sensor de toque)

Este sensor é utilizado para detecção de toques. Ele funciona como o tato do robô, sendo capaz de reconhecer quando o botão está pressionado, pulsado e liberado.

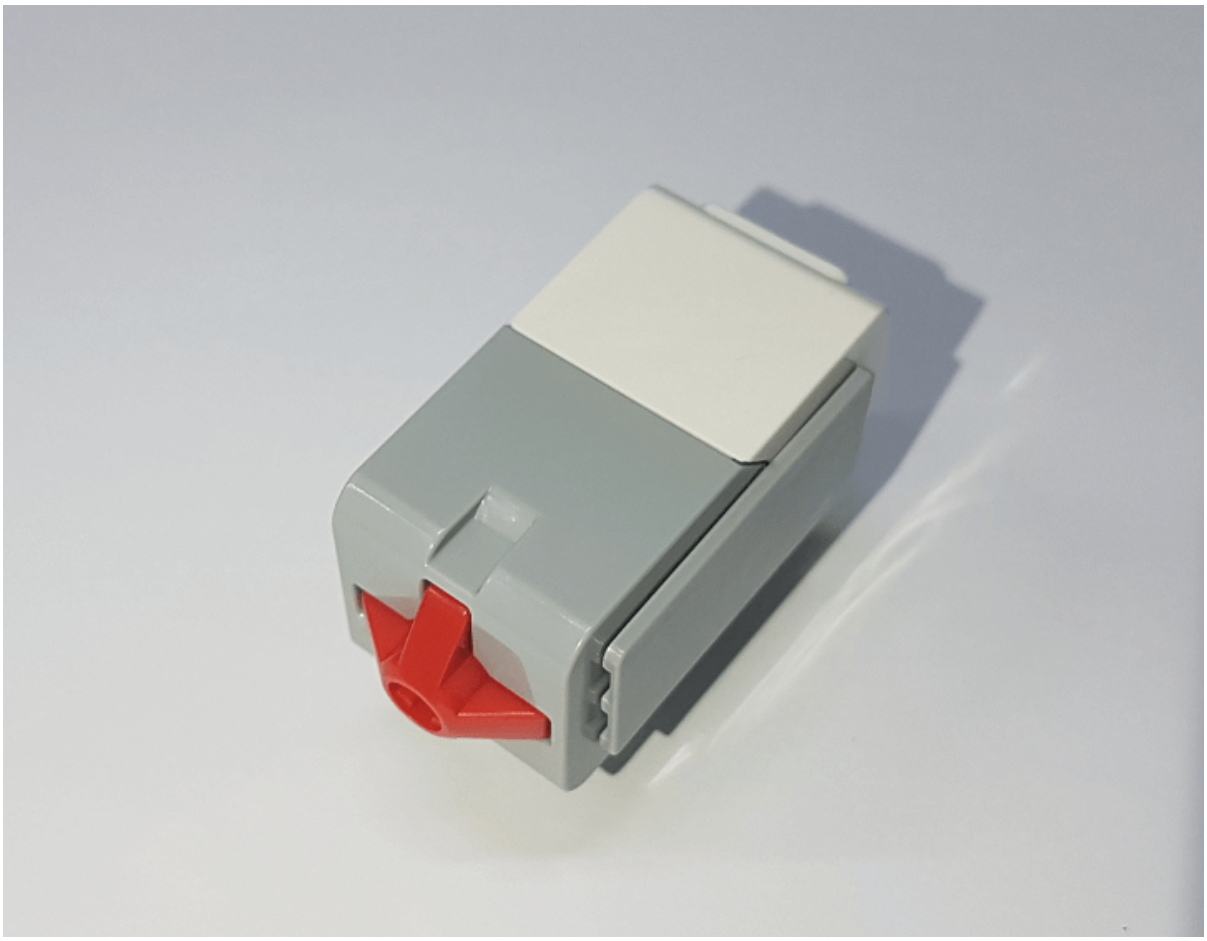


Figura 1.11: Touch Sensor (Sensor de toque)

## Gyro Sensor (Sensor de giro)

Este sensor é utilizado para medir o movimento e as mudanças na orientação da rotação do robô, e não está presente na versão Home Edition do EV3. Ele mede a taxa de rotação e a angulação em torno de seu próprio eixo.

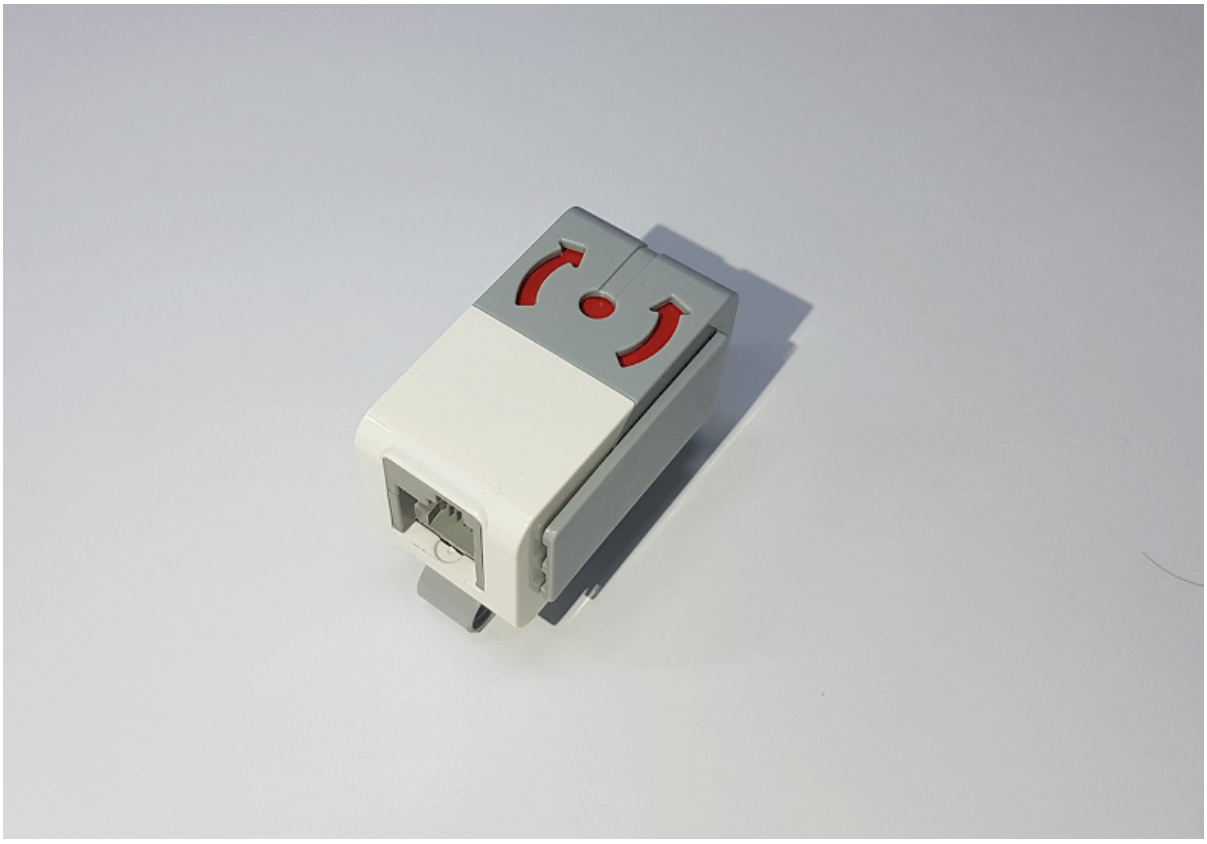


Figura 1.12: Gyro Sensor (Sensor de Giro)

## Colour Sensor (Sensor de cor)

O sensor de cor apresenta os seguintes modos de funcionamento: *Colour Mode* (Modo Cor) e *Light Intensity Mode* (Modo Luminosidade). Dessa forma, pode ser utilizado para detectar cores ou detectar luz.

- Colour Mode

Este modo reconhece sete cores: preto, marrom, verde, azul, amarelo, branco e vermelho, ainda possui o modo “sem cor”.

- Light Intensity Mode

Pode ser utilizado em um dos dois modos a seguir.

*Ambient Light Intensity Mode* (Modo de intensidade da luz ambiente): neste modo, o sensor mede a quantidade de luz presente no ambiente. A escala vai de 0 (muito escuro) a 100 (muito claro).

*Reflected Light Intensity Mode* (Modo de intensidade da luz refletida): neste modo, o sensor emite um feixe de luz vermelho e mede a quantidade refletida pela superfície ou objeto. A escala funciona da mesma forma que a do modo de luz ambiente.

> **Observação**
>
> Este livro faz uso do inglês britânico, pois optou-se pelo download do software (conforme será visto na parte seguinte do livro) no inglês britânico. Portanto, a palavra "Colour" é a versão britânica da palavra americana "Color".

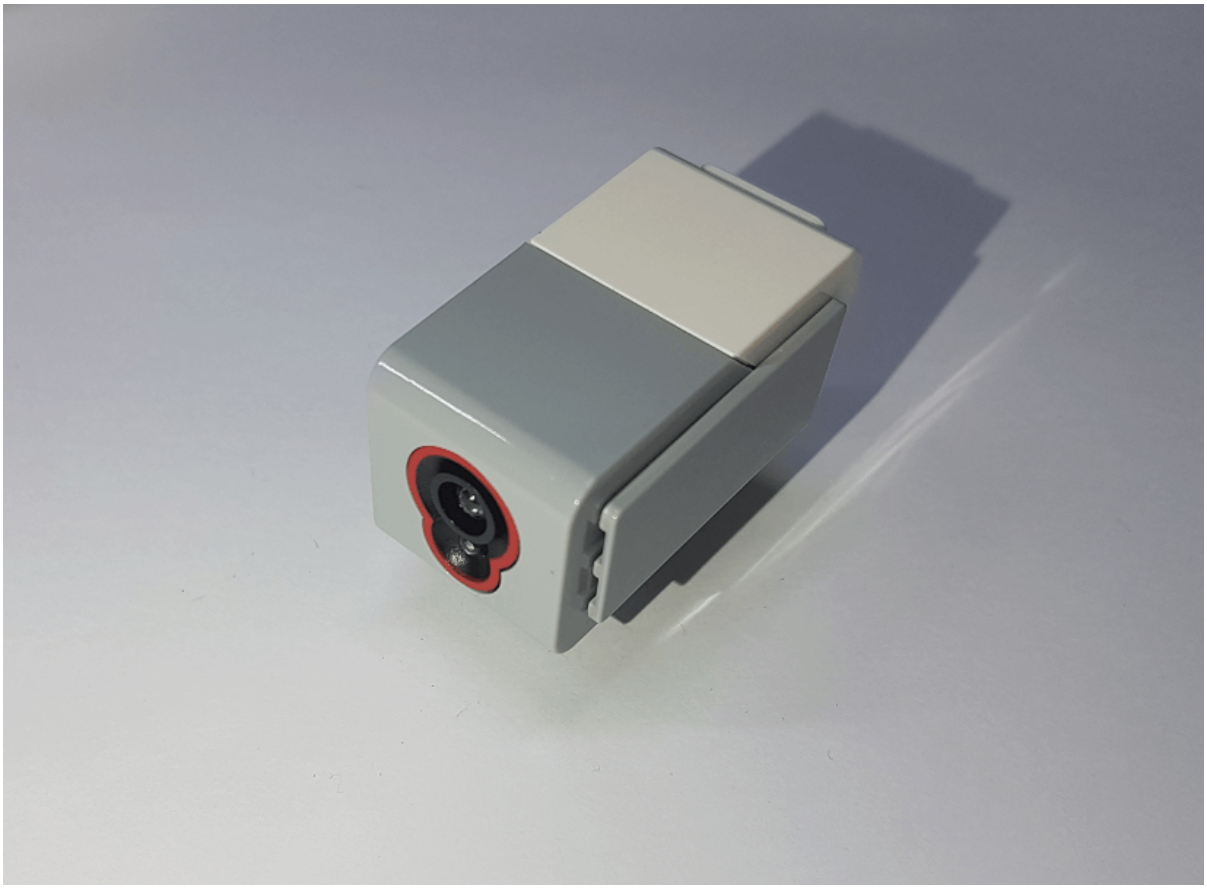


Figura 1.13: Colour Sensor (Sensor de cor)

## Ultrasonic Sensor (Sensor ultrassônico)

O sensor ultrassônico é utilizado para realizar medições de distâncias até um objeto que se encontra à sua frente. As medições podem ser realizadas em centímetros ou polegadas. O sensor possui alcance na faixa de 1 e 99 polegadas. Você pode realizar medições de 1 a 250 centímetros. Pode ser utilizado no modo de medição ou no modo de comparação. Os modos serão abordados mais profundamente na seção de programação.

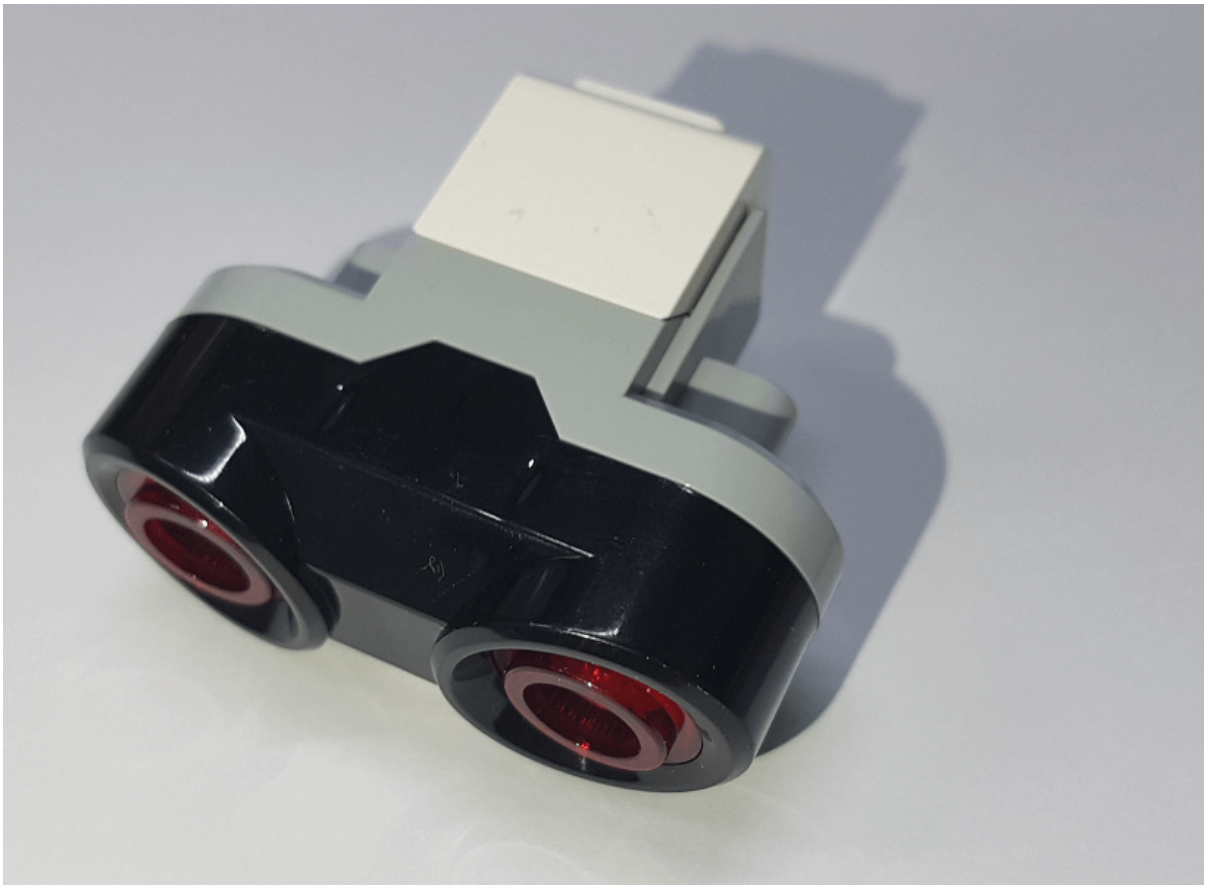


Figura 1.14: Ultrasonic Sensor (Sensor ultrassônico)

## Infrared Sensor (Sensor infravermelho) e Remote Infrared Beacon (Baliza remota infravermelho)

O sensor infravermelho e o Remote Infrared Beacon (que é o controle do sensor) estão presentes somente na versão Home Edition do EV3.

Este sensor detecta a luz infravermelha refletida em objetos e também detecta os sinais infravermelhos enviados por seu controle. Pode ser utilizado juntamente com o controle ou sem ele.

**FUNCIONAMENTO E EFICÁCIA DO SENSOR**

Algumas considerações devem ser feitas a respeito do funcionamento e eficácia do sensor infravermelho: a luz do sol pode afetar os sinais infravermelhos enviados e recebidos; sua eficácia está ligada à cor dos objetos que refletirão os sinais. Observa-se que objetos escuros refletem menos luz infravermelha que os objetos claros.

O Sensor Infravermelho apresenta 3 modos de funcionamento:

- Proximity Mode

Neste modo, o sensor envia um sinal infravermelho, e a partir da reflexão deste sinal pelo objeto à sua frente ele estima sua proximidade, que é definida em uma escala de 0 a 100, onde: 0 significa que o objeto está muito perto; 100 significa que o objeto está muito longe.

**OBSERVAÇÃO**

Diferente do sensor ultrassônico, que mede a distância, este sensor apenas oferece uma ideia de proximidade, o quão próximo o objeto se situa do sensor, tendo como base uma escala.

- Beacon Mode

Utilizado em conjunto com o Remote Infrared Control. Este modo é utilizado para a detecção da posição aproximada do controle em relação ao sensor por meio da recepção dos sinais infravermelhos vindos do controle. Além de oferecer a proximidade do controle, ele pode oferecer também a angulação em uma escala de –25 a 25 (em que os valores negativos correspondem ao lado esquerdo, e os positivos, ao lado direito).

**Observação**

Os valores de angulação e proximidade dependem da força do sinal infravermelho, por isso esses valores são aproximados, e podem não corresponder exatamente aos valores reais.

- Remote Mode

O Remote Infrared Beacon possui 4 canais e 5 botões, constituindo um controle remoto para o sensor. O sensor é capaz de identificar qual botão foi pressionado e em qual canal o controle está operando.

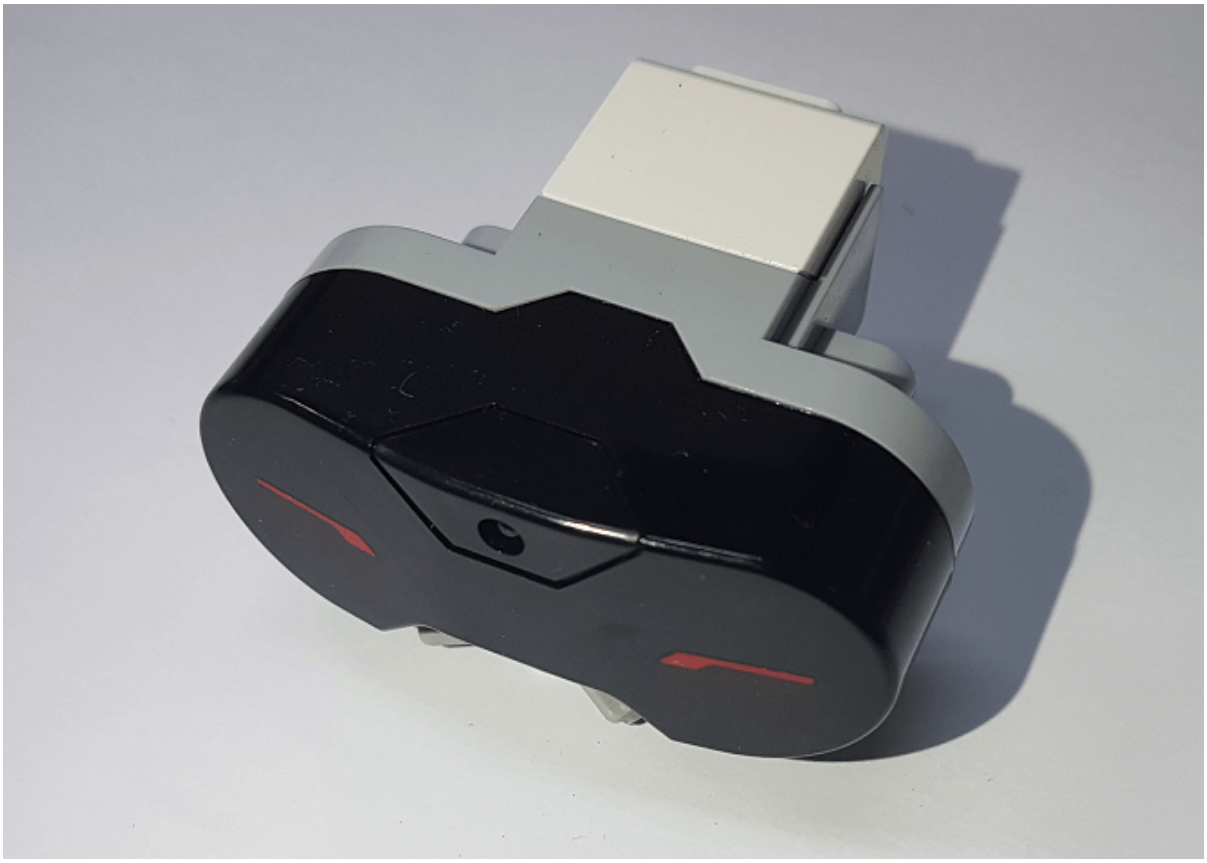


Figura 1.15: Infrared Sensor (Sensor infravermelho)

Figura 1.16: Remote Infrared Beacon (Controle do sensor infravermelho)

## 1.3 Qual versão comprar?

Se você já possui alguma experiência com os kits robóticos da LEGO®, a melhor opção para você talvez seja a versão Education, pois com ela encontrará mais possibilidades e liberdade de criação.

A versão Home Edition é destinada principalmente para as crianças. Os próprios modelos de robôs exemplificam bem isto. Os modelos disponibilizados para a construção possuem um design chamativo, que atrai a atenção das crianças. Isso é observável também nos modelos de robôs, que funcionam a partir do controle feito pelo

usuário por meio do controle infravermelho e seu sensor receptor. E o software desta versão possui menos recursos.

Quanto aos softwares, você consegue fazer o download do software de ambas as versões de forma totalmente gratuita. Logo, isso não é um empecilho ou fator que conte muito na escolha da versão mais adequada para você.

Já na questão de componentes dos kits, há uma notável diferença. A versão Education conta com uma bateria recarregável e isto é um grande diferencial. A bateria facilita em muito sua vida, pois basta conectar um carregador ao Brick e pronto, na maioria dos casos, não será preciso que você fique desmontando seus robôs. Além disso, a versão Education conta com mais sensores, o que lhe confere mais possibilidades.

Para uso educacional, a versão Education continua sendo a melhor escolha, pois conta com uma aplicação mais voltada para o ensino de ciências. No próprio site da LEGO® há manuais e conteúdos de aulas para professores e educadores. É importante ressaltar que esse material se encontra disponível apenas em inglês. Caso você deseje participar de competições, sem dúvida nenhuma, a versão Education é a melhor escolha, pois conta com bateria recarregável e mais sensores.

## Conclusão

Finaliza-se se aqui a primeira parte do livro. Esta parte abordou questões do hardware, fornecendo ao leitor um conhecimento sobre os componentes do kit, bem como suas utilidades e modos de funcionamento.

A seguir vem a segunda parte do livro, que tratará o software (LEGO® MINDSTORMS® Education EV3 ou a versão Home Basic). Esta parte contempla desde a instalação do software até as ferramentas disponibilizadas nele (as *Tools*). Caso o leitor possua o kit NXT, este próximo capítulo mostra que é possível utilizar o

software EV3 para o seu kit. Há um capítulo em que abordaremos como realizar os procedimentos adequados para a utilização do software EV3 para o kit NXT.

# O SOFTWARE

Esta é a parte 2 do livro, ela contempla questões acerca do software EV3, tanto a versão Home Edition, quanto a Education. Se você ainda não fez download do ambiente de programação EV3, esta parte pode ser imprescindível para você.

Para os leitores que possuem o kit anterior, o NXT 2.0, saibam que é possível utilizar o novo software. Esta parte também será de muita importância, pois ensina como realizar os procedimentos necessários para a utilização do kit NXT 2.0 com o novo software.

Além disso, será abordada a aba *Tools*, presente no software. A aba *Tools* consiste em uma série que recursos que podem ser utilizados para a construção de um programa. Porém, o principal é que é nela que se encontra a ferramenta utilizada para o update do firmware do Brick. Manter o firmware atualizado é fundamental.

## Capítulo 2
## Instalando o software LEGO® MINDSTORMS® EV3 Home Edition

Para instalar o software de programação basta clicar no link:

https://www.lego.com/en-us/mindstorms/downloads/download-software

A seguinte tela será exibida. Nela, escolha para qual sistema operacional será feito o download (Windows ou OS X). Basta clicar no botão que corresponde ao seu sistema operacional.

**NOTA**

O software LEGO® MINDSTORMS® Education EV3 também possui sua versão App, disponível para Android (compatível com tablets) e iOS (compatível com iPad). Vale ressaltar que os Apps não são ferramentas tão robustas quanto os softwares.

Android - https://play.google.com/store/apps/details?id=com.lego.education.ev3

iOS - https://itunes.apple.com/app/lego-mindstorms-education/id959374369

mindstorms
MINDSTORMS PRODUCT APPS BUILD A ROBOT FAN ROBOTS LEARN TO PROGRAM VIDEOS GALLERY OLD GALLERY GAMES
DOWNLOADS
EV3 SOFTWARE DOWNLOAD (PC/MAC)
Download for Windows
Download for OS X
PROGRAM YOUR ROBOTS WITH THE ICON-BASED SOFTWARE

Figura 2.1: Instalação da versão Home Edition

Depois de concluído o download, basta localizar o arquivo e executá-lo. Uma tela será mostrada para confirmar a instalação, bastando clicar em “sim”. Em seguida, os arquivos constituintes do software serão extraídos.

Após a extração o instalador será carregado.

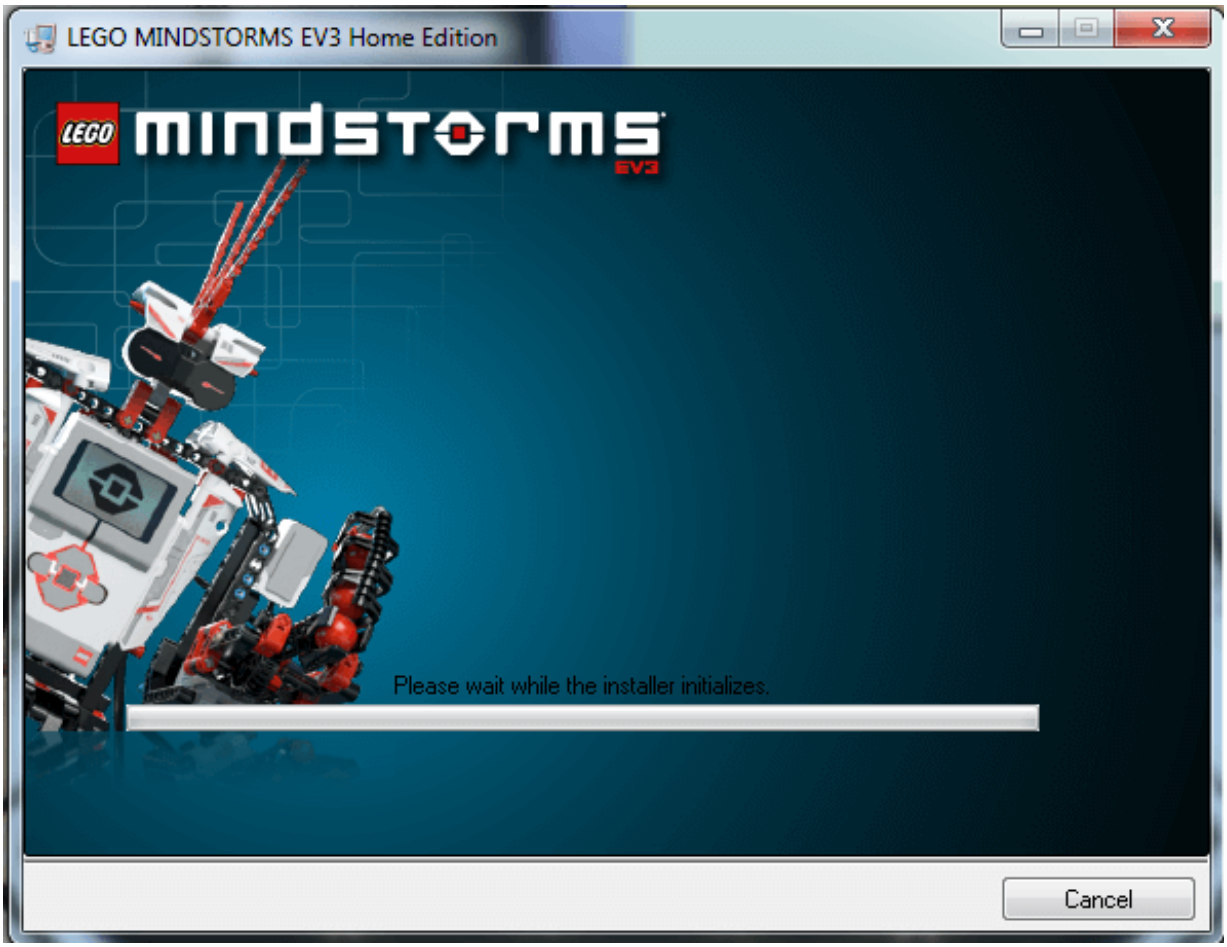



Figura 2.2: Instalação da versão Home Edition

Um local específico pode ser escolhido para a instalação, ou o usuário pode simplesmente seguir em frente, e instalar automaticamente. Feito isso, pressiona-se o botão “Next”.

O próximo passo consiste em aceitar os termos da licença, conforme as seguintes telas.

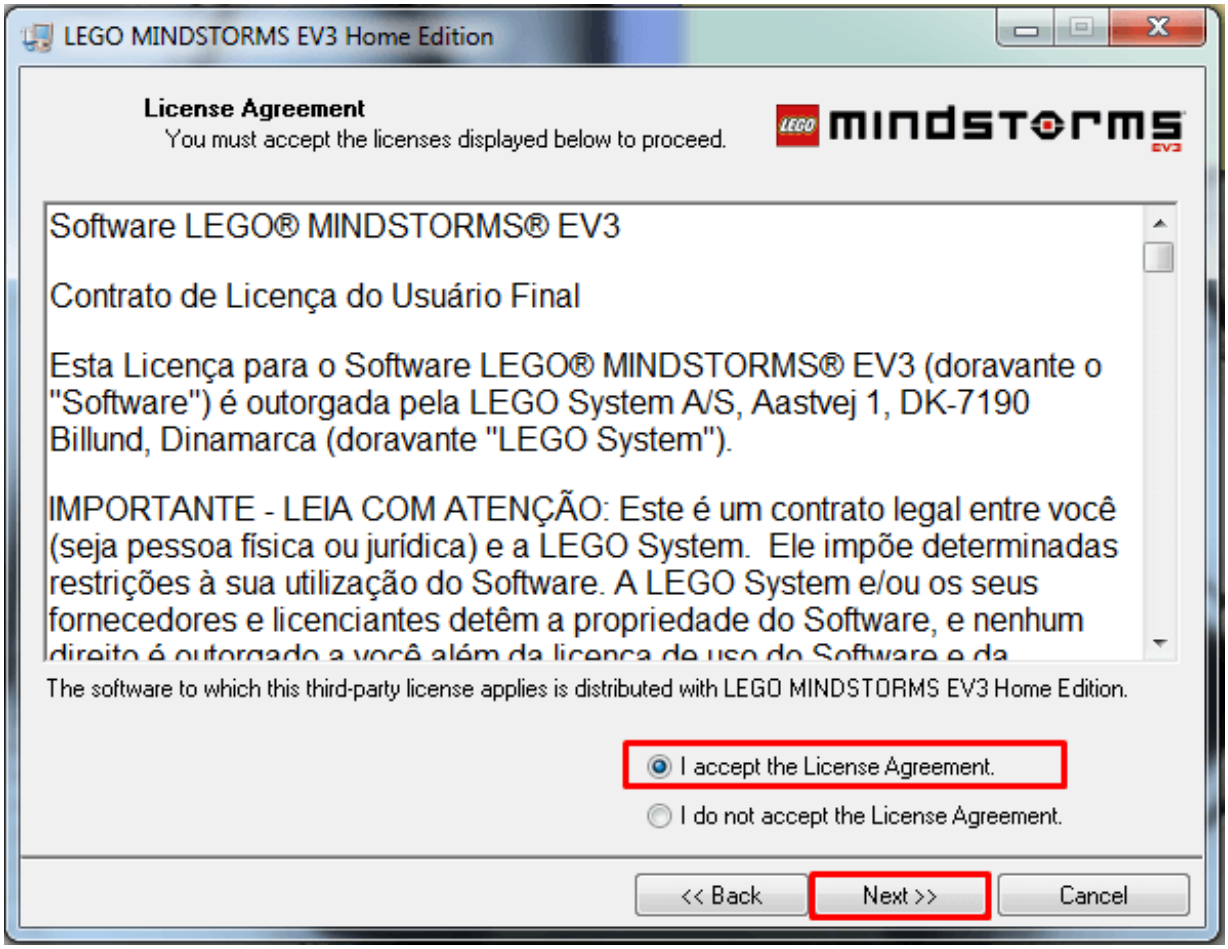

LEGO MINDSTORMS EV3 Home Edition
License Agreement
You must accept the licenses displayed below to proceed.
mindstorms
Software LEGO® MINDSTORMS® EV3
Contrato de Licença do Usuário Final
Esta Licença para o Software LEGO® MINDSTORMS® EV3 (doravante o "Software") é outorgada pela LEGO System A/S, Aastvej 1, DK-7190 Billund, Dinamarca (doravante "LEGO System").
IMPORTANTE - LEIA COM ATENÇÃO: Este é um contrato legal entre você (seja pessoa física ou jurídica) e a LEGO System. Ele impõe determinadas restrições à sua utilização do Software. A LEGO System e/ou os seus fornecedores e licenciantes detêm a propriedade do Software, e nenhum
The software to which this third-party license applies is distributed with LEGO MINDSTORMS EV3 Home Edition.
I accept the License Agreement.
I do not accept the License Agreement.
<< Back
Next >>
Cancel

Figura 2.3: Instalação da versão Home Edition

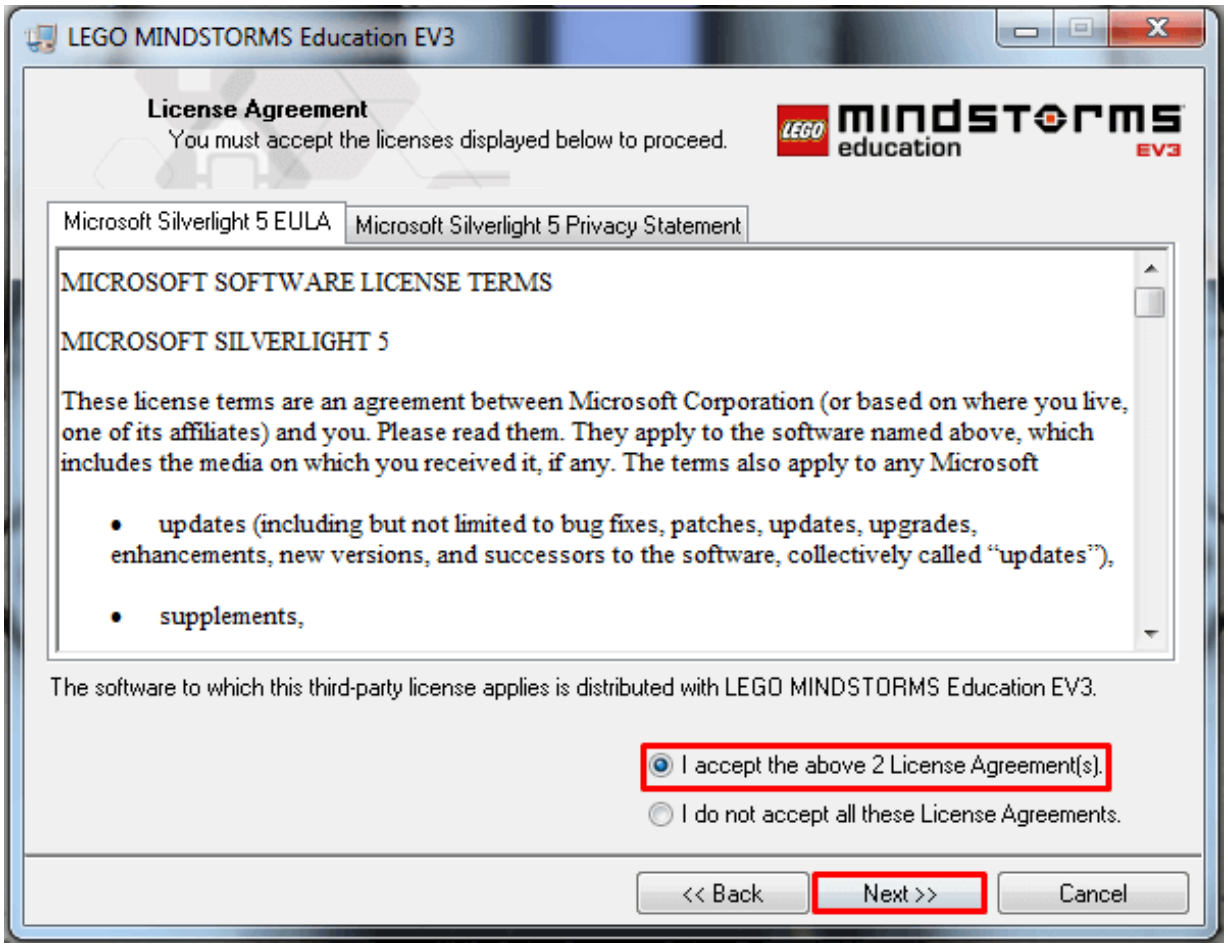

LEGO MINDSTORMS Education EV3
License Agreement
You must accept the licenses displayed below to proceed.
mindstorms education EV3
Microsoft Silverlight 5 EULA
Microsoft Silverlight 5 Privacy Statement
MICROSOFT SOFTWARE LICENSE TERMS
MICROSOFT SILVERLIGHT 5
These license terms are an agreement between Microsoft Corporation (or based on where you live, one of its affiliates) and you. Please read them. They apply to the software named above, which includes the media on which you received it, if any. The terms also apply to any Microsoft
updates (including but not limited to bug fixes, patches, updates, upgrades, enhancements, new versions, and successors to the software, collectively called "updates"),
supplements,
The software to which this third-party license applies is distributed with LEGO MINDSTORMS Education EV3.
I accept the above 2 License Agreement(s).
I do not accept all these License Agreements.
<< Back
Next >>
Cancel

Figura 2.4: Instalação da versão Home Edition

A partir dessa etapa, a instalação estará em progresso.

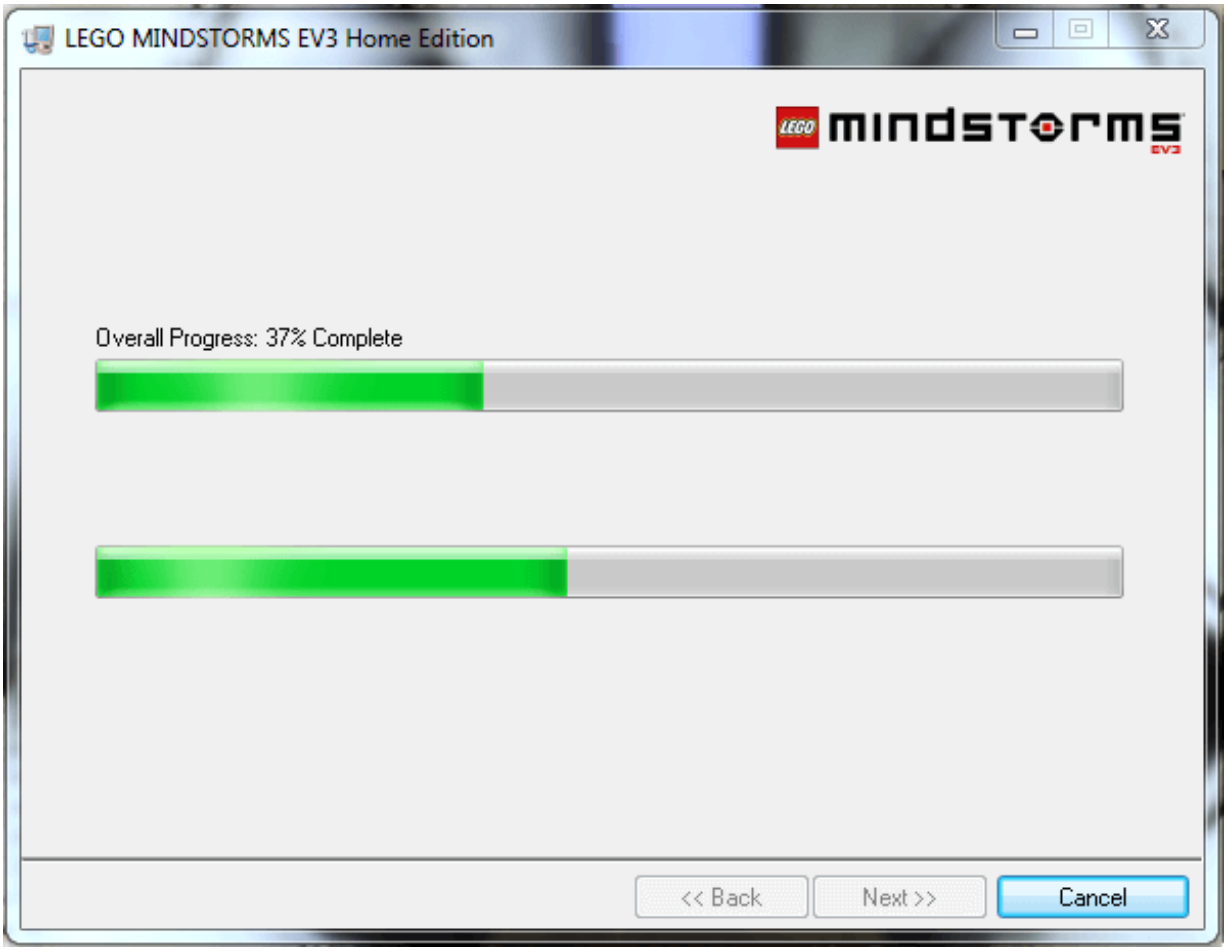



Figura 2.5: Instalação da versão Home Edition

Com a conclusão, a próxima tela é mostrada, bastando agora clicar em “Finish”. Assim, o LEGO® MINDSTORMS® Home Edition está instalado e pronto para uso.

Figura 2.6: Instalação da versão Home Edition - Ícone de Instalação

Agora é só clicar no ícone do software e esperar até que ele seja inicializado. Está é a tela inicial do software:

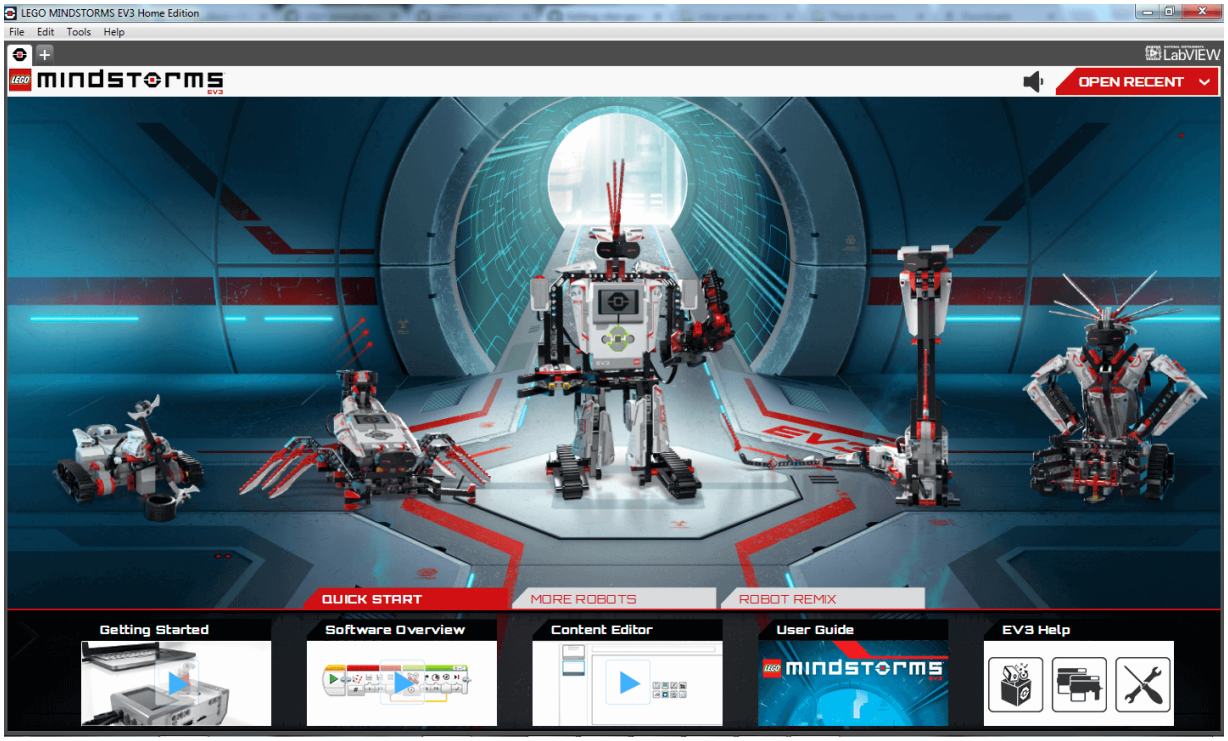



Figura 2.7: Tela inicial

# Capítulo 3
# Instalando o software LEGO® MINDSTORMS® Education EV3

Para instalar o software de programação basta clicar no link:

https://education.lego.com/en-us/downloads/mindstorms-ev3

Antes de clicar em download, deve-se selecionar o sistema operacional utilizado no computador do usuário. Também é possível realizar a mudança do idioma do software para o português. Feito isso, é só clicar em “Download".

education
PRÉ ESCOLA
ELEMENTAR
ENSINO FUNDAMENTAL
fazer compras
Transferências
Apoio, suporte
LEGO® MINDSTORMS® Education EV3 Downloads
Baixe seu software
Escolha Plataforma / Dispositivo
Windows (7, 8.1, 10)
Escolha a sua língua
Inglês (Reino Unido)
DOWNLOAD
849 MB
Se você tem algum produto LEGO MINDSTORMS Education EV3, então você precisa baixar este software. Inclui recursos de professores, uma ferramenta de documentação, registro de dados, instruções de construção e tutoriais.
Clique aqui para saber mais sobre os diferentes Windows 10 vers ...
EXIBIR REQUISITOS DO SISTEMA
iOS

Figura 3.1: Instalação da versão Education

A seguir, abre-se o arquivo baixado e pressiona-se o botão “Sim” para executá-lo.

O próximo passo dá início à extração do arquivo.

Pode-se escolher um local específico para a instalação ou simplesmente seguir em frente e instalar automaticamente. Feito isso, pressiona-se o botão “Next” e depois novamente em “Next”.

Na próxima tela (mostrada a seguir) escolhe-se a versão do software EV3. A versão *Teacher Edition* foi feita para os professores. Com ela, os professores podem criar projetos que não podem ser modificados: é possível bloqueá-los e ainda é possível esconder algo. Na *Student Version* não é possível fazer isto. Aqui é o espaço destinado à escolha da versão do software que melhor se adeque às funcionalidades desejadas pelo usuário.

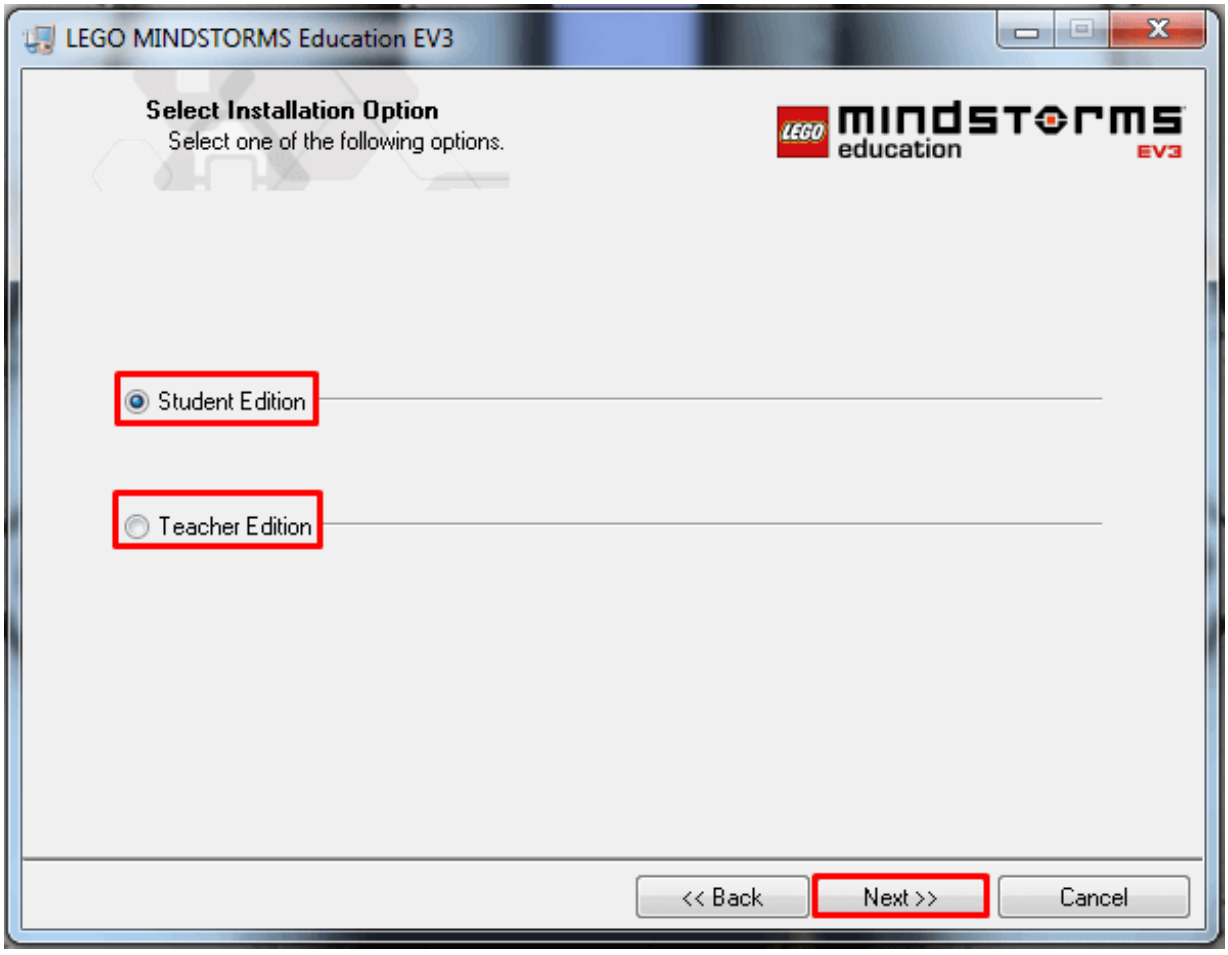



Figura 3.2: Instalação da versão Education

Agora é só aceitar os termos de licença que serão mostrados.

LEGO MINDSTORMS Education EV3
License Agreement
You must accept the licenses displayed below to proceed.
mindstorms education EV3
Software LEGO® MINDSTORMS® EV3
Contrato de Licença do Usuário Final
Esta Licença para o Software LEGO® MINDSTORMS® EV3 (doravante o "Software") é outorgada pela LEGO System A/S, Aastvej 1, DK-7190 Billund, Dinamarca (doravante "LEGO System").
IMPORTANTE - LEIA COM ATENÇÃO: Este é um contrato legal entre você (seja pessoa física ou jurídica) e a LEGO System. Ele impõe determinadas restrições à sua utilização do Software. A LEGO System e/ou os seus fornecedores e licenciantes detêm a propriedade do Software, e nenhum
The software to which this third-party license applies is distributed with LEGO MINDSTORMS Education EV3.
I accept the License Agreement.
I do not accept the License Agreement.
<< Back
Next >>
Cancel

Figura 3.3: Instalação da versão Education

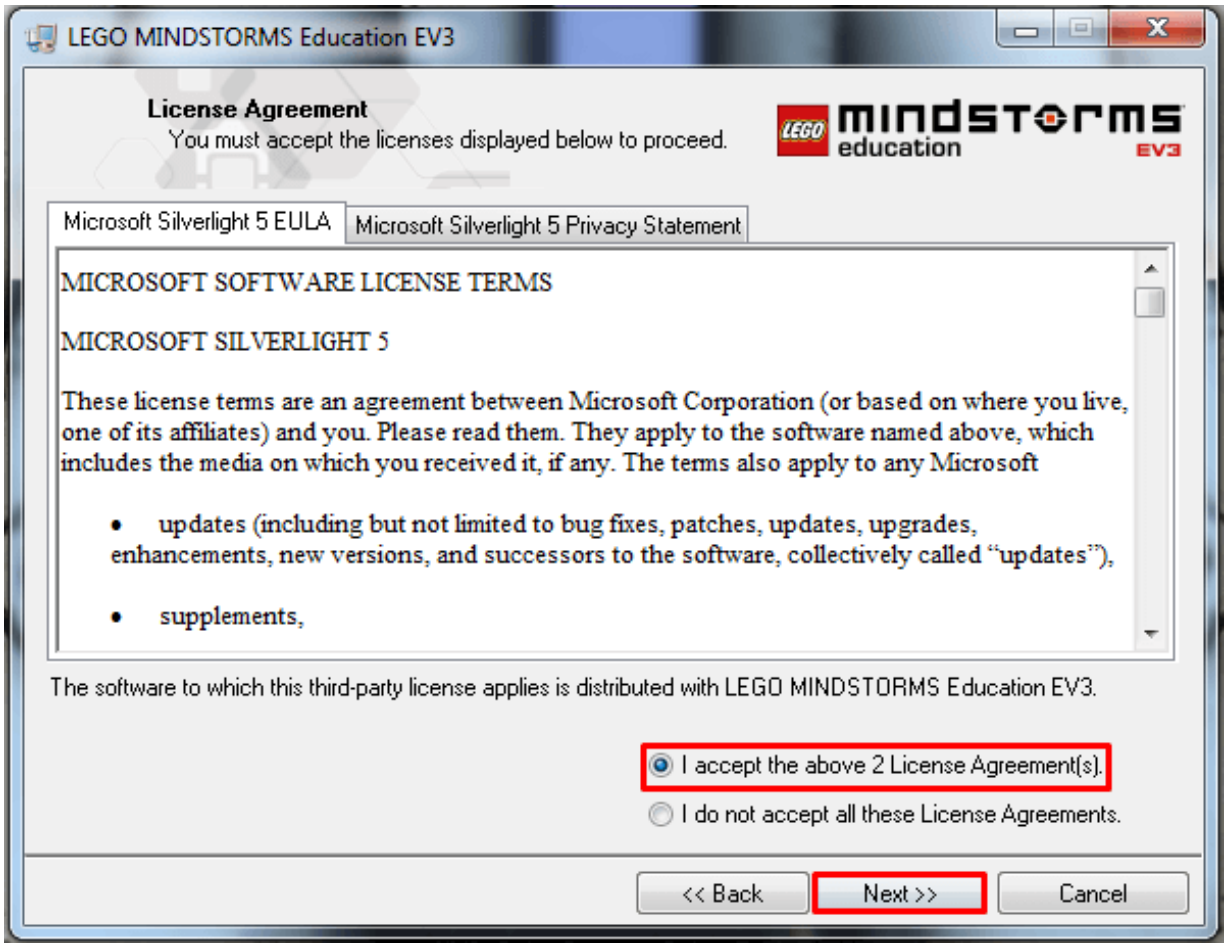

LEGO MINDSTORMS Education EV3
License Agreement
You must accept the licenses displayed below to proceed.
mindstorms education EV3
Microsoft Silverlight 5 EULA
Microsoft Silverlight 5 Privacy Statement
MICROSOFT SOFTWARE LICENSE TERMS
MICROSOFT SILVERLIGHT 5
These license terms are an agreement between Microsoft Corporation (or based on where you live, one of its affiliates) and you. Please read them. They apply to the software named above, which includes the media on which you received it, if any. The terms also apply to any Microsoft
updates (including but not limited to bug fixes, patches, updates, upgrades, enhancements, new versions, and successors to the software, collectively called "updates"),
supplements,
The software to which this third-party license applies is distributed with LEGO MINDSTORMS Education EV3.
I accept the above 2 License Agreement(s).
I do not accept all these License Agreements.
<< Back
Next >>
Cancel

Figura 3.4: Instalação da versão Education

A instalação estará em progresso.

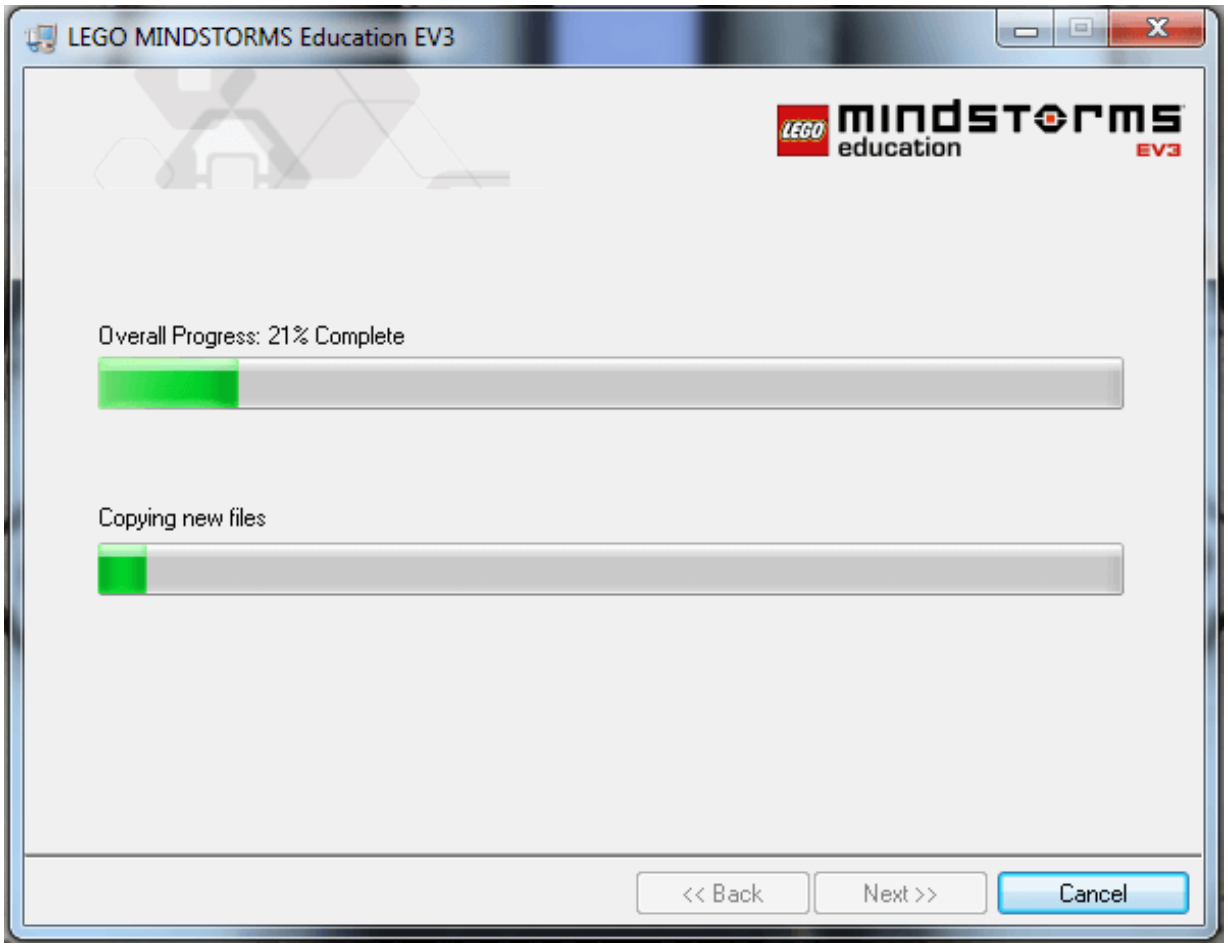



Figura 3.5: Instalação da versão Education

Com a conclusão é só clicar em “Finish”. Dessa forma, o LEGO® MINDSTORMS® Education EV3 estará instalado e pronto para uso.

O ícone a seguir é criado na área de trabalho.

Figura 3.6: Instalação da versão Education - Ícone de Instalação

Agora é só clicar no ícone do software e esperar até que ele seja inicializado. Esta é a tela inicial do software:

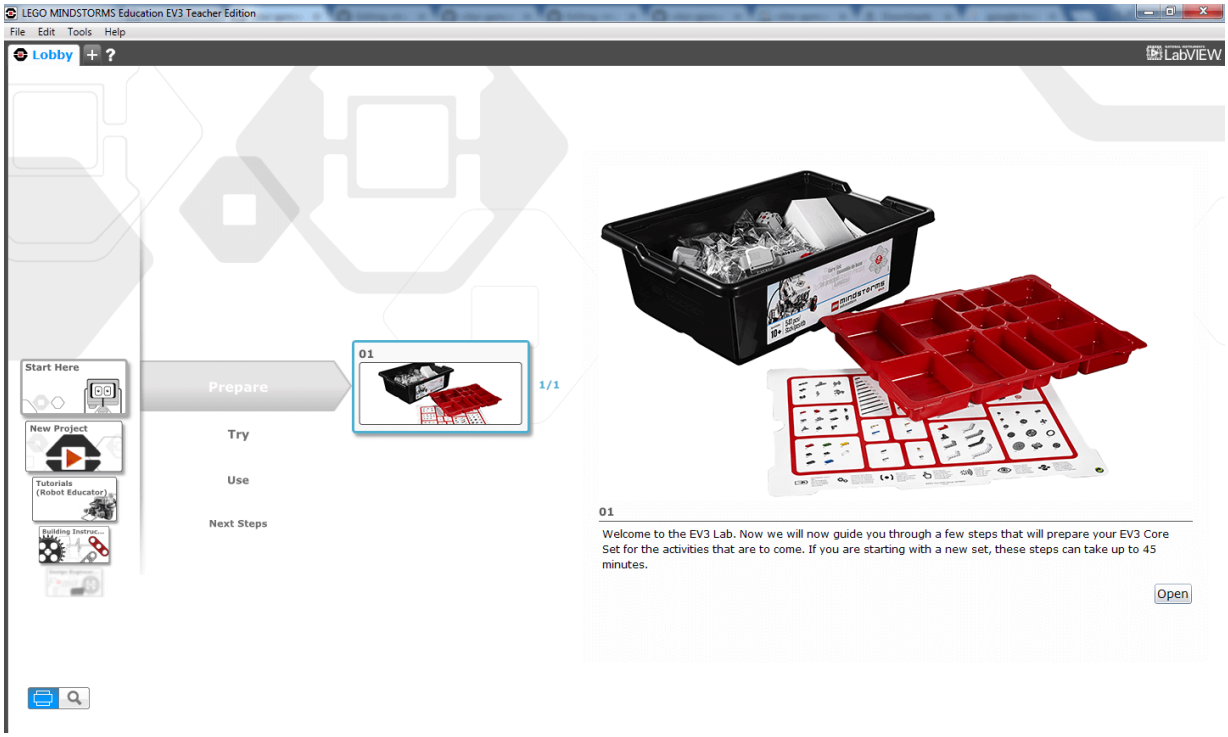

LEGO MINDSTORMS Education EV3 Teacher Edition
File Edit Tools Help
Lobby
LabVIEW
Start Here
New Project
Tutorials (Robot Educator)
Prepare
Try
Use
Next Steps
01
1/1
01
Welcome to the EV3 Lab. Now we will now guide you through a few steps that will prepare your EV3 Core Set for the activities that are to come. If you are starting with a new set, these steps can take up to 45 minutes.
Open

Figura 3.7: Tela inicial

# Capítulo 4
# O software LEGO® MINDSTORMS® EV3

## 4.1 Home Edition

Ao iniciar o LEGO® MINDSTORMS® EV3 Home Edition, aparecerá a seguinte tela:

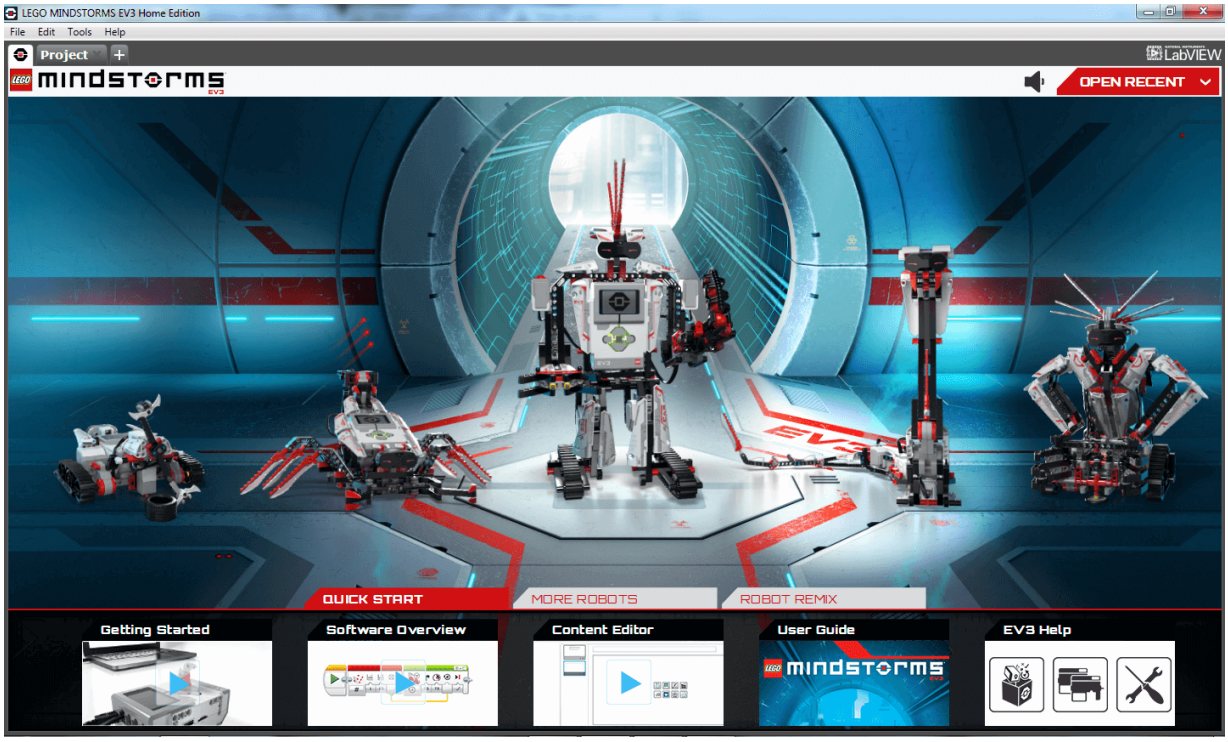



Figura 4.1: LEGO® MINDSTORMS® EV3 Home Edition - Área Inicial

Essa área é o *Lobby*; nela estão as informações necessárias para os primeiros passos no software. Vale destacar para os usuários que estão tendo contato pela primeira vez com uma linguagem visual a aba *Quick Start*, que oferece uma visão geral do kit. Os cinco robôs exibidos na maior parte da tela fazem parte das *Robot Missions*.

### Quick Start

O *Quick Start* é perfeito para novos usuários, aqueles que não possuem conhecimento prévio sobre programação e o kit. Esta aba oferece uma ótima visão geral sobre o software.

Figura 4.2: LEGO® MINDSTORMS® EV3 Home Edition - Quick Start

## Robot Missions

No *Lobby*, há cinco robôs, os quais fazem parte das *Robot Missions*. São cinco robôs com instruções de programação e montagem disponíveis. Ao passar o mouse sobre um robô, é possível ver informações e a opção de construir e programar. As *Robot Missions* constituem uma ótima forma de aprendizagem inicial.

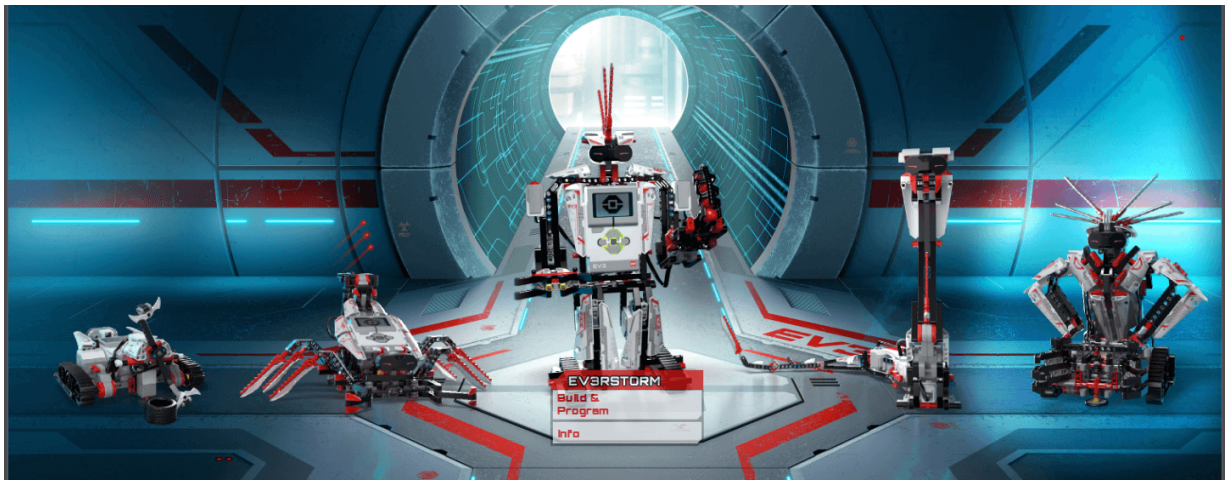



Figura 4.3: LEGO® MINDSTORMS® EV3 Home Edition - Principais robôs das Robot Missions

Clicando sobre o robô é possível obter mais detalhes acerca de suas funcionalidades.

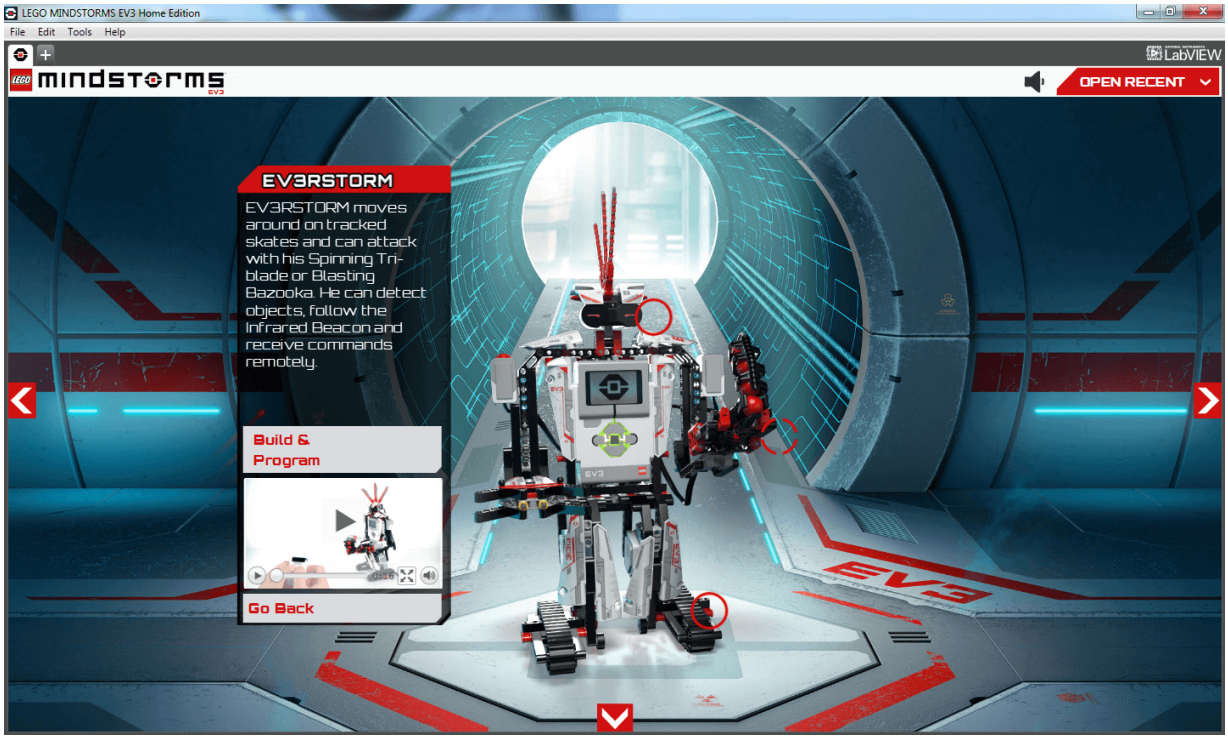

LEGO MINDSTORMS EV3 Home Edition
File Edit Tools Help
LabVIEW
mindstorms
OPEN RECENT
EV3RSTORM
EV3RSTORM moves around on tracked skates and can attack with his Spinning Tri-blade or Blasting Bazooka. He can detect objects, follow the Infrared Beacon and receive commands remotely.
Build & Program
Go Back
EV3

Figura 4.4: LEGO® MINDSTORMS® EV3 Home Edition - Escolhendo um robô

## 4.2 Education

Recentemente, o software LEGO® MINDSTORMS® Education EV3 sofreu uma atualização. Agora, ele está na versão 1.4.0. Houve algumas mudanças significativas no *Lobby*, quanto à organização e o conteúdo apresentado. O *Lobby* apresenta mais recursos e ficou ainda melhor.

### O que há de novo?

Uma mudança notável foi quanto ao conteúdo e informações disponibilizados nessa nova versão. Com a atualização, o software oferece tudo que você precisa, desde como desembalar e organizar as peças e conteúdo do kit, até como fazer seus primeiros programas.

Com as melhorias e aperfeiçoamentos feitos no *Lobby*, o usuário tem uma curva de aprendizado ainda maior. Por meio dessas mudanças, o usuário novo e inexperiente pode se sentir mais à vontade e logo se familiarizará com o kit. Vamos conhecer um pouco mais das mudanças apresentadas nessa nova versão.

Essa é a nova área inicial do software, o *Lobby*:

LEGO MINDSTORMS Education EV3 Teacher Edition
File Edit Tools Help
Lobby
LabVIEW
Start Here
New Project
Tutorials (Robot Educator)
Building Instruc...
Prepare
Try
Use
Next Steps
Start here!
We are excited to help you to get started with using LEGO MINDSTORMS Education EV3. Let us guide you through a couple of steps that will enable you to start using it in your classroom.

Figura 4.5: Novo Lobby

Nela oferecem-se as seguintes opções:

## Para a Student Edition

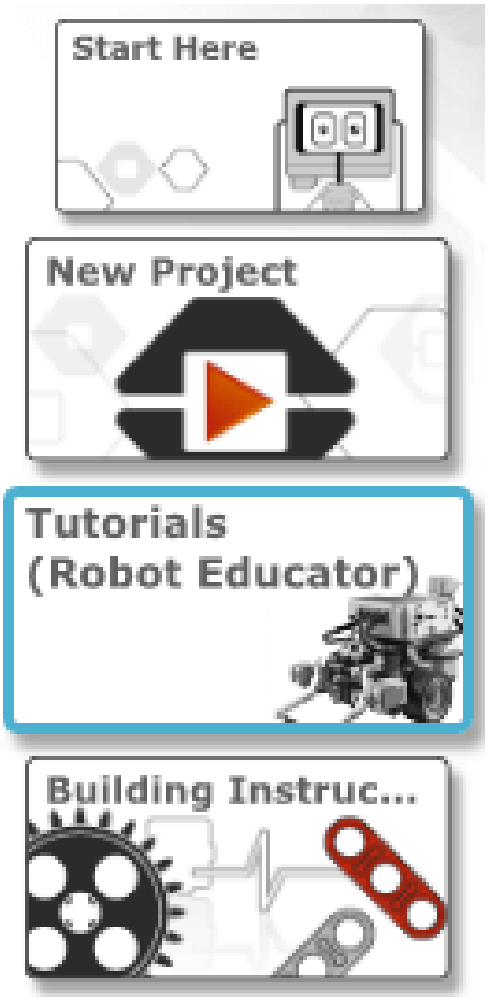



Figura 4.6: Novo Lobby - Recursos

- *Start Here* – É ideal para os usuários novos, que estão tendo o contato com o kit pela primeira vez. Aqui você encontrará desde de instruções para preparar o seu kit (desembalando e organizando as peças), até como criar os seus primeiros programas.
- *New Project* – Clicando aqui você pode escolher criar um novo projeto ou um novo experimento.
- *Tutorials (Robot Educator)* – Oferece uma seleção de tutoriais para que você possa conhecer melhor o seu kit. Nele você encontrará noções básicas acerca do hardware e de blocos de programação.

- *Building Instructions* – Disponibiliza instruções de montagem para diversos protótipos, desde protótipos mais simples, envolvendo apenas um sensor e o Brick, até instruções de montagem para uma base robótica (*Building Ideas*). Há também as opções *Core Set Models*, com modelos de robôs que podem ser criados a partir do kit principal e ainda a opção *Expansion Set Models*, com tutoriais de montagem para os usuários que possuem o kit de Expansão (este kit conta com peças adicionais de montagem). Por último, na opção *Element Survey*, você encontra uma lista com os componentes do kit, tanto para a versão *Core Set*, quanto para a *Expansion Set*.

**Para a Teacher Edition**

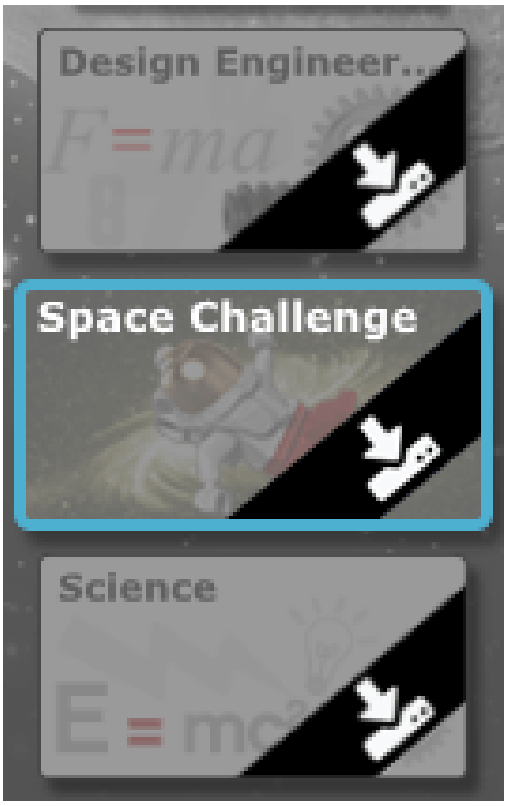



Figura 4.7: Novo Lobby - Recursos

A Teacher Edition ainda conta com recursos adicionais, como:

- *Designer Engineer*

- *Space Challenge*
- *Science*

Para utilizar tais recursos, você só precisa baixá-los, pelo link disponibilizado:

https://education.lego.com/en-us/downloads/mindstorms-ev3/curriculum.

É importante ressaltar que alguns destes recursos foram desenvolvidos para serem utilizados com os kits de expansão como, o EV3 *Space Challenge Set* (https://education.lego.com/en-us/products/ev3-space-challenge-set/45570) e o EV3 *Science Add-On Pack* (https://education.lego.com/en-us/products/ev3-science-add-on-pack/5005270). Estes são recursos voltados principalmente para os professores que ministram disciplinas que abrangem STEM, pois fornecem aplicações de cunho científico que podem ser utilizadas em sala de aula.

> **STEM**
>
> STEM é uma sigla internacional para as áreas de Ciências, Tecnologia, Engenharia e Matemática.

**Show Overlay Help**

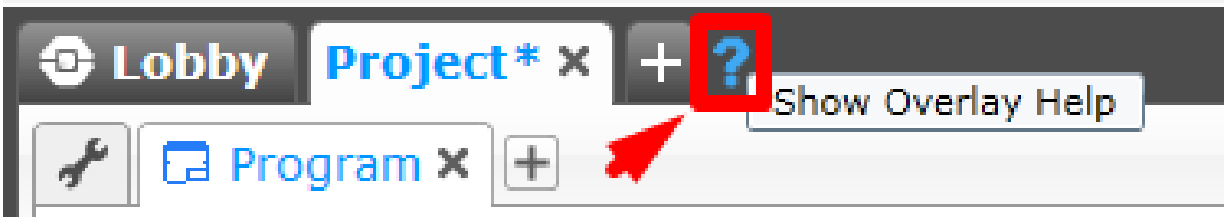



Figura 4.8: Ícone do Show Overlay Help

Além dessa mudança no *Lobby*, outras funcionalidades e recursos foram acrescentados no software. Podemos destacar a

funcionalidade *Show Overlay Help*, representada pelo ícone de interrogação, que aparece sempre posterior à aba do projeto. Sua utilização é bem simples, porém, ela é um recurso muito útil para quem está começando. Portanto, se você está começando ou possui alguma dúvida sobre o seu ambiente de desenvolvimento, é só clicar no símbolo de interrogação.

Ao clicar no ícone, será exibida na tela uma imagem contendo informações sobre cada parte (recursos e funcionalidades) que constitui o ambiente de desenvolvimento.

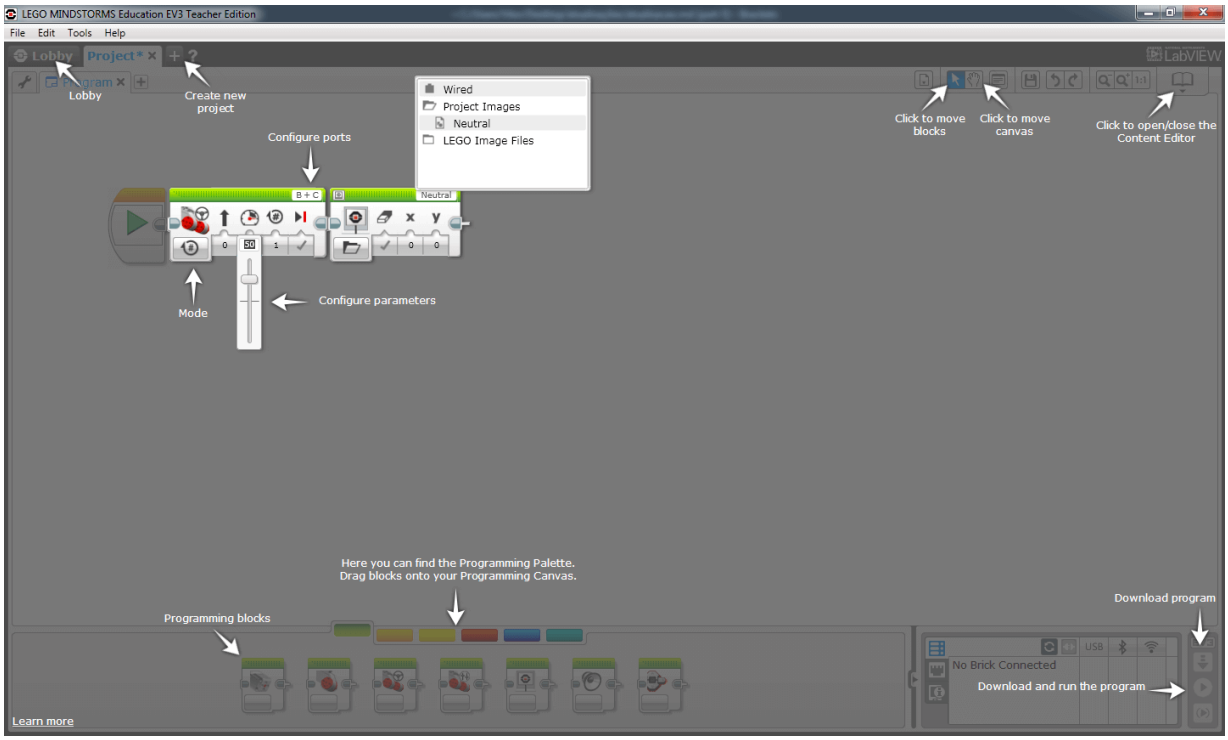



Figura 4.9: Novo Lobby

## Tools

Houve também uma mudança na aba *Tools*. Ela agora conta com algumas ferramentas novas. Uma ferramenta muito interessante, que foi adicionada, é a *Present Port View*.

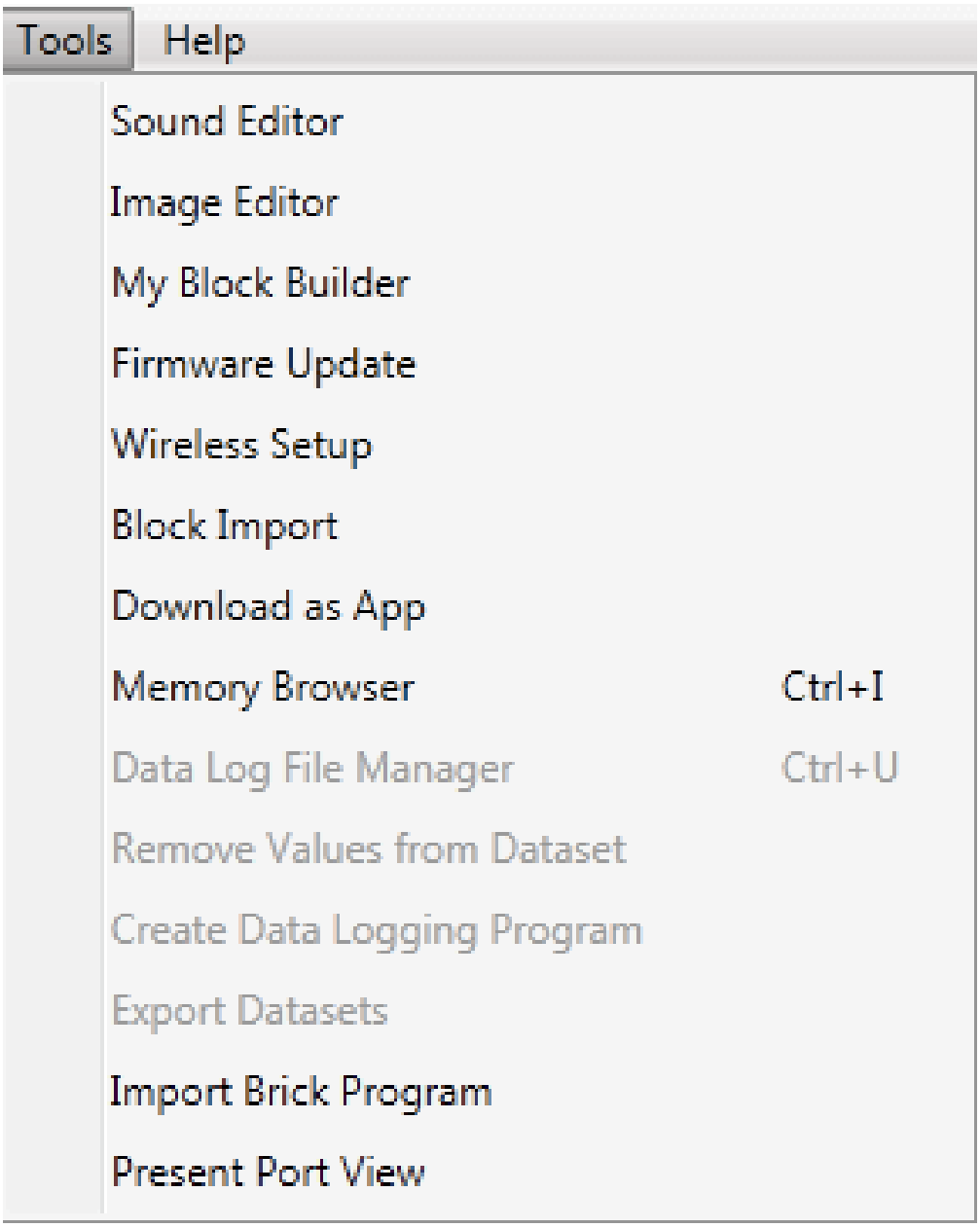



Figura 4.10: Aba Tools

Com a ferramenta *Present Port View*, você consegue identificar diretamente, em sua máquina, quais componentes estão conectados ao Brick (sensores e motores) e, caso estejam conectados, em quais portas eles estão conectados.

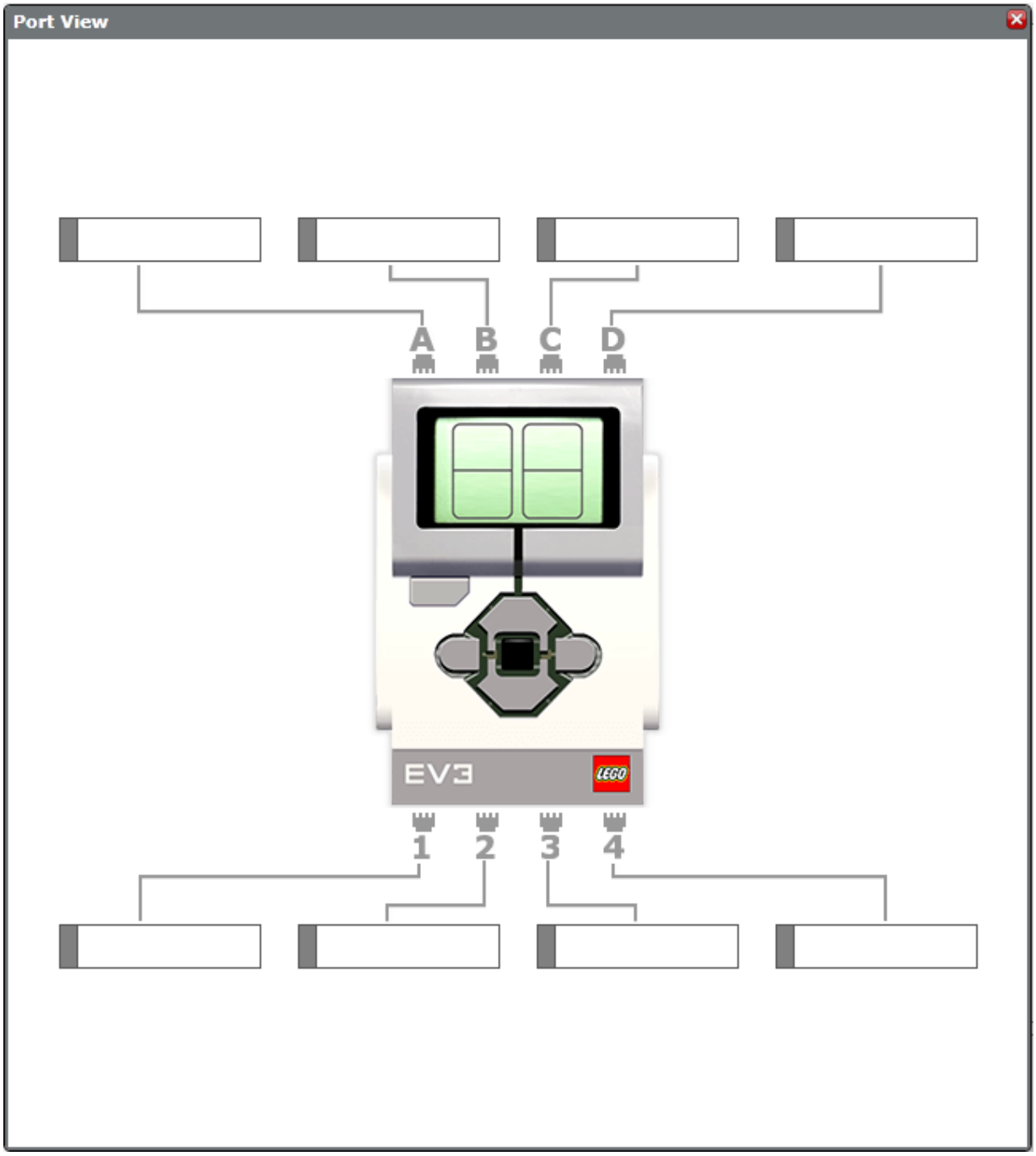



Figura 4.11: Present Port View

## A versão anterior do software LEGO® MINDSTORMS® Education EV3

Para que você possa notar as diferenças mais de perto, e de forma mais eficaz, esta seção faz uma apresentação da versão anterior. Como a atualização ainda é recente, pode ser que alguns usuários ainda não tenham atualizado o seu software. Por isso, aqui você perceberá, de forma clara, as mudanças realizadas na nova versão.

Ao iniciar o LEGO® MINDSTORMS® Education EV3, a seguinte tela era exibida.

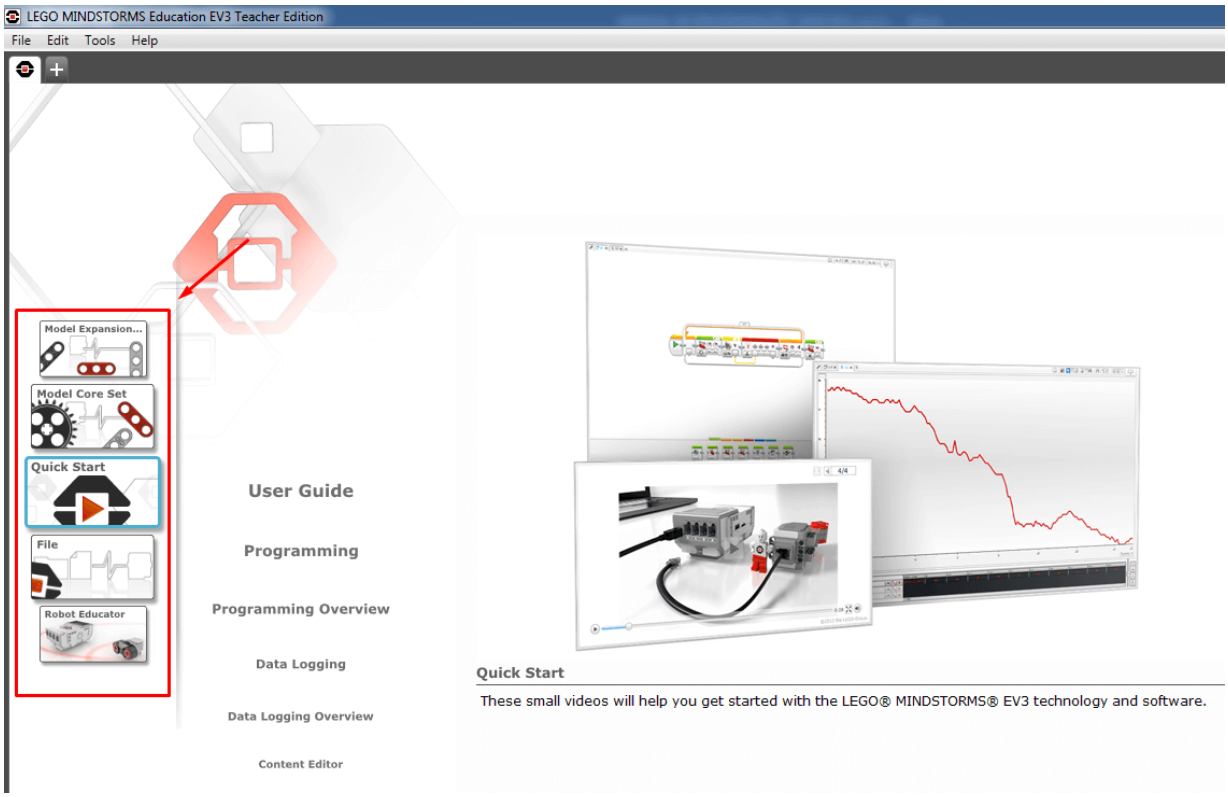



Figura 4.12: LEGO® MINDSTORMS® Education EV3 - Área Inicial

No canto esquerdo do software encontrava-se uma aba com cinco opções:

## Model Expansion Set

Apresenta modelos mais elaborados de robôs que podem ser construídos, oferecendo instruções de montagem e funcionamento. São seis modelos de robôs: Tank Bot, Znap, Stair Climber, Elephant,

Spinner Factory e Remote Control. Aqui também se encontra uma lista com todas as peças LEGO® disponíveis no mercado.

> **Observação**
>
> Os robôs apresentados no *Model Expansion Set* requerem o kit de expansão para a sua montagem. Para mais informações, acesse:
>
> https://education.lego.com/en-us/products/lego-mindstorms-education-ev3-expansion-set/45560.

### Model Core Set

Oferece quatro modelos de robô para construção. Esses robôs são mais complexos e oferecem aplicações práticas, além de expandir seu conhecimento de programação, pois utilizam blocos de programação avançados. Os robôs são os seguintes: Gyro Boy, Colour Sorter, Puppy e o Robot Arm H25.

### Quick Start

Este é o referencial para quem está começando. Nesta aba, encontram-se tutoriais com vídeos de curta duração sobre o software, a programação, o funcionamento dos blocos etc. Essa é uma ótima oportunidade para quem quer entender um pouco mais sobre o software.

### File

Apresenta três opções:

- *Open Project* – Para abrir um projeto já existente;
- *New Project* – Para começar um novo projeto;
- *Open Recent* – Para abrir o último projeto executado, ou seja, o mais recente.

### Robot Educator

Esta área é voltada para os educadores. Contém instruções de montagem que vão desde os modelos mais simples até modelos um pouco mais complexos de robôs. Encontram-se aqui informações de hardware, ferramentas, *data logging* e um guia para professores.

## 4.3 Mudança de idioma

O software LEGO® MINDSTORMS® Education EV3 permite que você realize mudança de idioma. Isso é bem simples de fazer, basta acessar o site, escolher o sistema operacional adequado, selecionar o idioma e baixar o software novamente. Após isso, você deve instalar o software (conforme fez da primeira vez.)

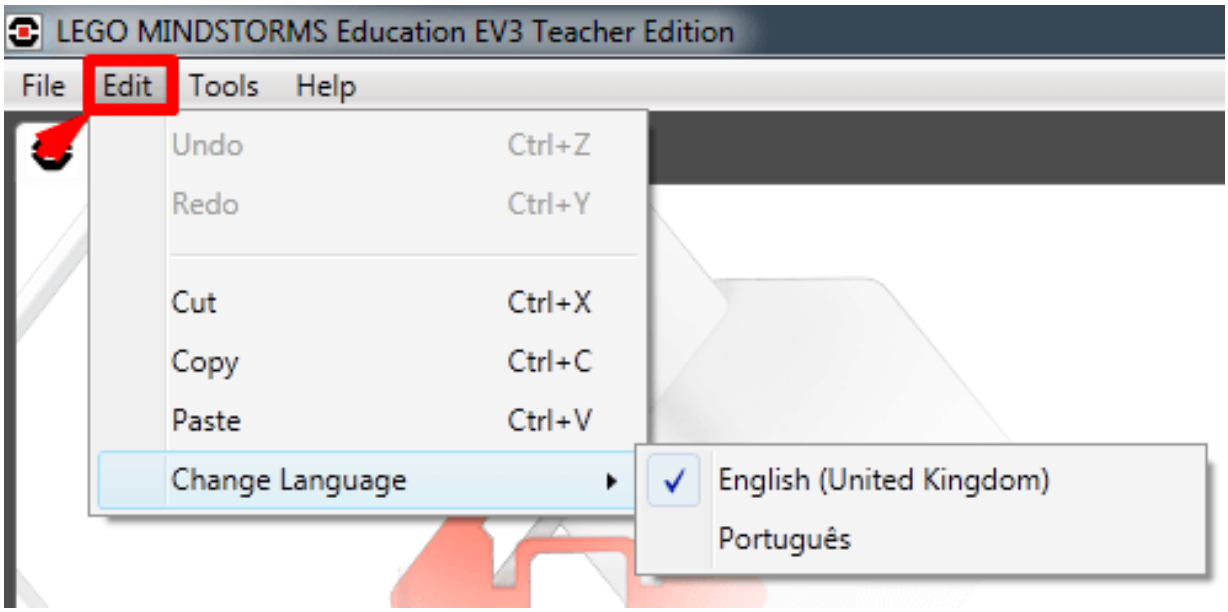



Figura 4.13: Alterando a linguagem

Depois de ter feito o download e ter instalado o software, abra o seu LEGO® MINDSTORMS® Education EV3, clique em `Edit` , escolha a opção *Change Language* e selecione o idioma desejado. Feito isso, uma mensagem aparecerá pedindo que você reinicie o programa para que as mudanças possam surtir efeito. Você pode alterar o

idioma sempre que quiser, mas para completar a alteração é preciso reiniciar o software.

## Conclusão

Até agora foram vistos detalhes importantes sobre o funcionamento e as funcionalidades do software. Estes capítulos são fundamentais para que você, leitor, tenha um bom entendimento de sua ferramenta de programação.

# Capítulo 5
# Utilizando o software EV3 no NXT 2.0

O antecessor do EV3 é o kit NXT 2.0, que ainda é muito utilizado. O NXT 2.0 é muito mais simples e modesto que o EV3, por ser de uma geração anterior. As diferenças entre o EV3 e o NXT 2.0 não são apenas visuais. O EV3 conta com um poder de processamento muito maior, sensores mais modernos, e uma série de outras vantagens que já foram ditas aqui. Porém, mesmo tendo sido substituído pelo EV3, ele ainda se mostra uma ótima ferramenta para a aprendizagem de robótica.

O NXT 2.0 também pode ser usado com o software EV3. Logo, se você possui o kit NXT 2.0 também pode realizar projetos com o software EV3. Mas a mudança de plataforma é uma ótima escolha, tendo em vista que o software EV3 é um ambiente moderno e recente, que facilita a criação de novos programas, pois conta com inúmeras melhorias. Esses aprimoramentos são um dos fatores que tornam conveniente a migração para o software EV3.

É importante ressaltar que alguns dos blocos do EV3 não funcionam para o NXT 2.0. Quando se tenta utilizá-los aparece um sinal de alerta.

Os blocos são:

- *Medium Motor*
- *Brick Status Light*
- *Infrared Sensor*
- *Gyro Sensor*
- *Array Operations*
- *Data Logging*
- *Messaging*
- *Bluetooth Connection*
- *Invert Motor*

**Observação**

Apesar de o bloco *Gyro Sensor* não ser compatível com o NXT 2.0, isso não impossibilita a utilização do sensor de giro presente no kit NXT 2.0. Para isso, basta utilizar um outro bloco que realiza a mesma função, porém, próprio e compatível com o kit: o bloco *RWW Gyro Sensor*. Para isso, basta fazer o download do bloco *Gyro Sensor* no seguinte link: https://www.lego.com/en-us/mindstorms/downloads e importá-lo para seu software (isto será visto mais adiante).

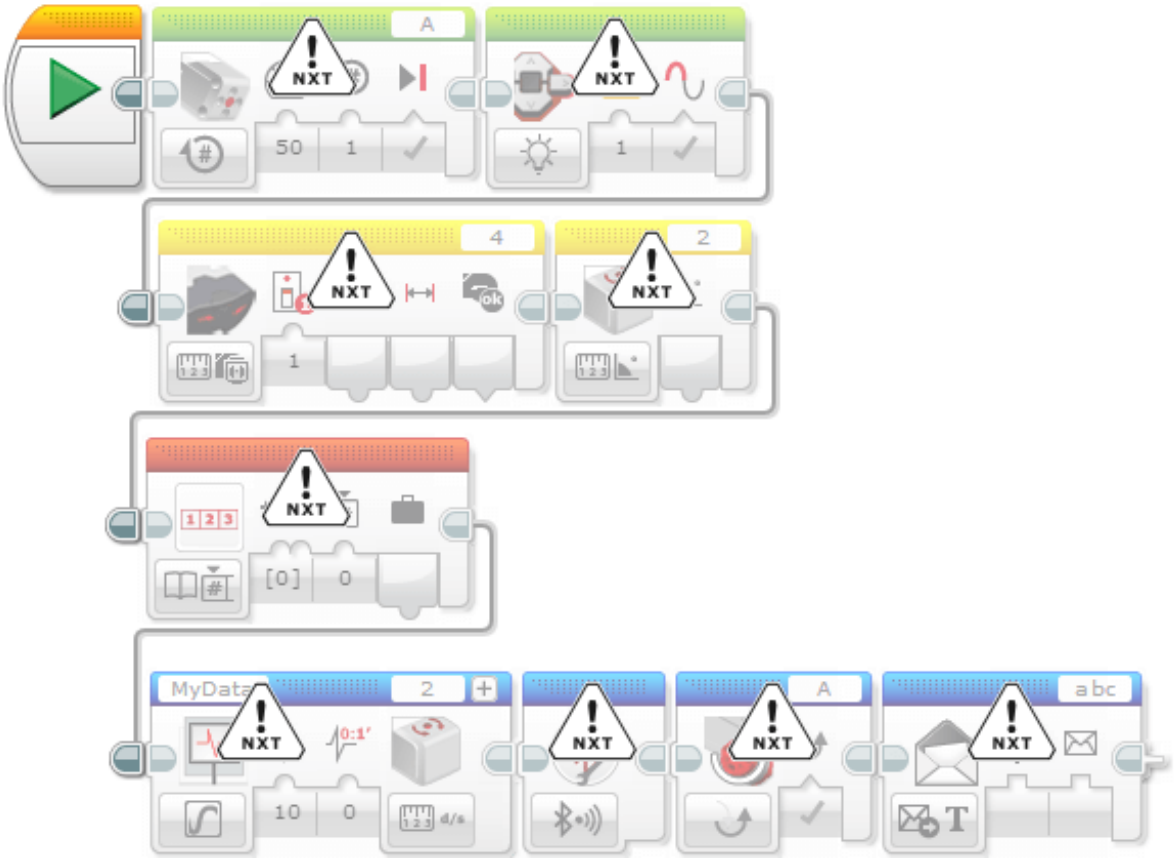



Figura 5.1: Blocos incompatíveis com o NXT

Um fator de caráter estético, porém notável no EV3, é o seu visual. O software está com uma interface mais moderna que o seu

antecessor. Isso pode até ajudar em uma melhor compreensão do código, já que se trata de uma linguagem visual.

O software EV3 apresenta algum grau de automação dos blocos, conforme observado em *Move Tank* e *Move Steering*. Esses blocos realizam respectivamente a função de andar em linha reta e a de fazer curvas. No LEGO® MINDSTORMS® NXT 2.0 estas funções deviam ser criadas pelo próprio usuário utilizando os blocos disponíveis, uma vez que não existiam blocos específicos para tal.

Para utilizar o software EV3 no NXT é preciso realizar antes alguns passos. O primeiro é o **update do Firmware**.

É necessário que o Firmware do brick seja atualizado, mas essa atualização deve ser feita no próprio software, o LEGO® MINDSTORMS® NXT 2.0.

Para realizar o update, conecta-se o robô ao PC e abre-se o software LEGO® MINDSTORMS® NXT 2.0.

Na aba *Tools*, clica-se em *Update NXT Firmware...*

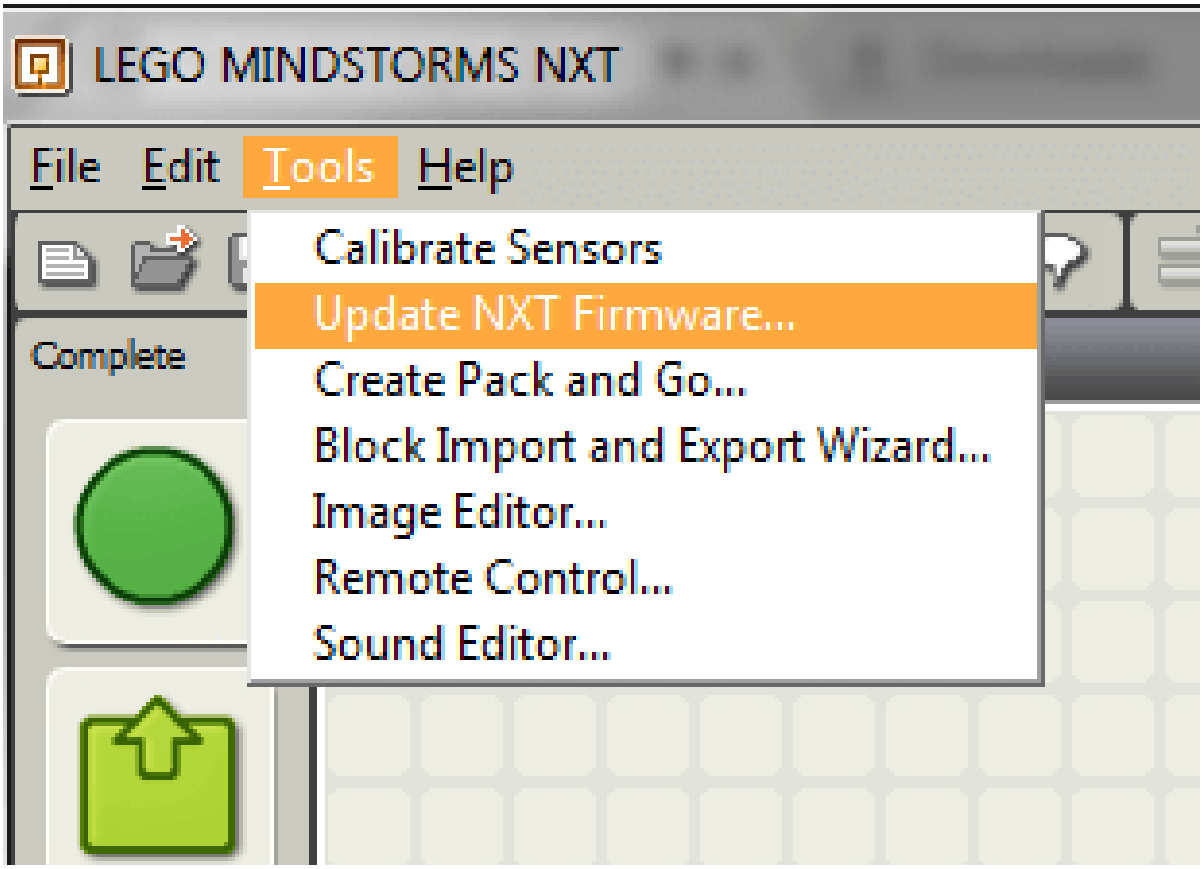



Figura 5.2: Realizando o update - passo 1

Em seguida, pressiona-se o botão *Download*.

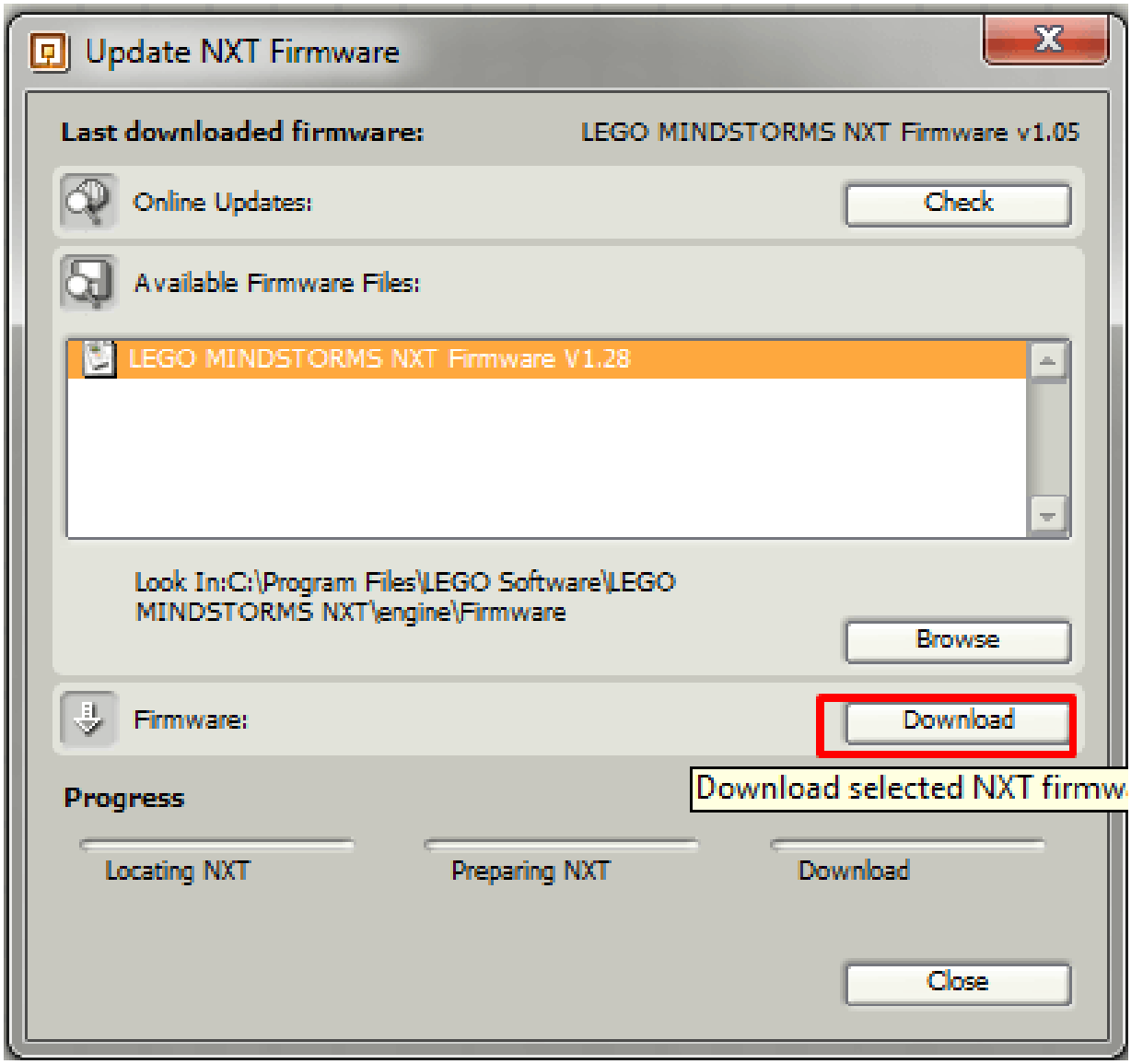



Figura 5.3: Realizando o update - passo 2

Feito isso, o Firmware está atualizado, de modo que é possível encerrar o software agora.

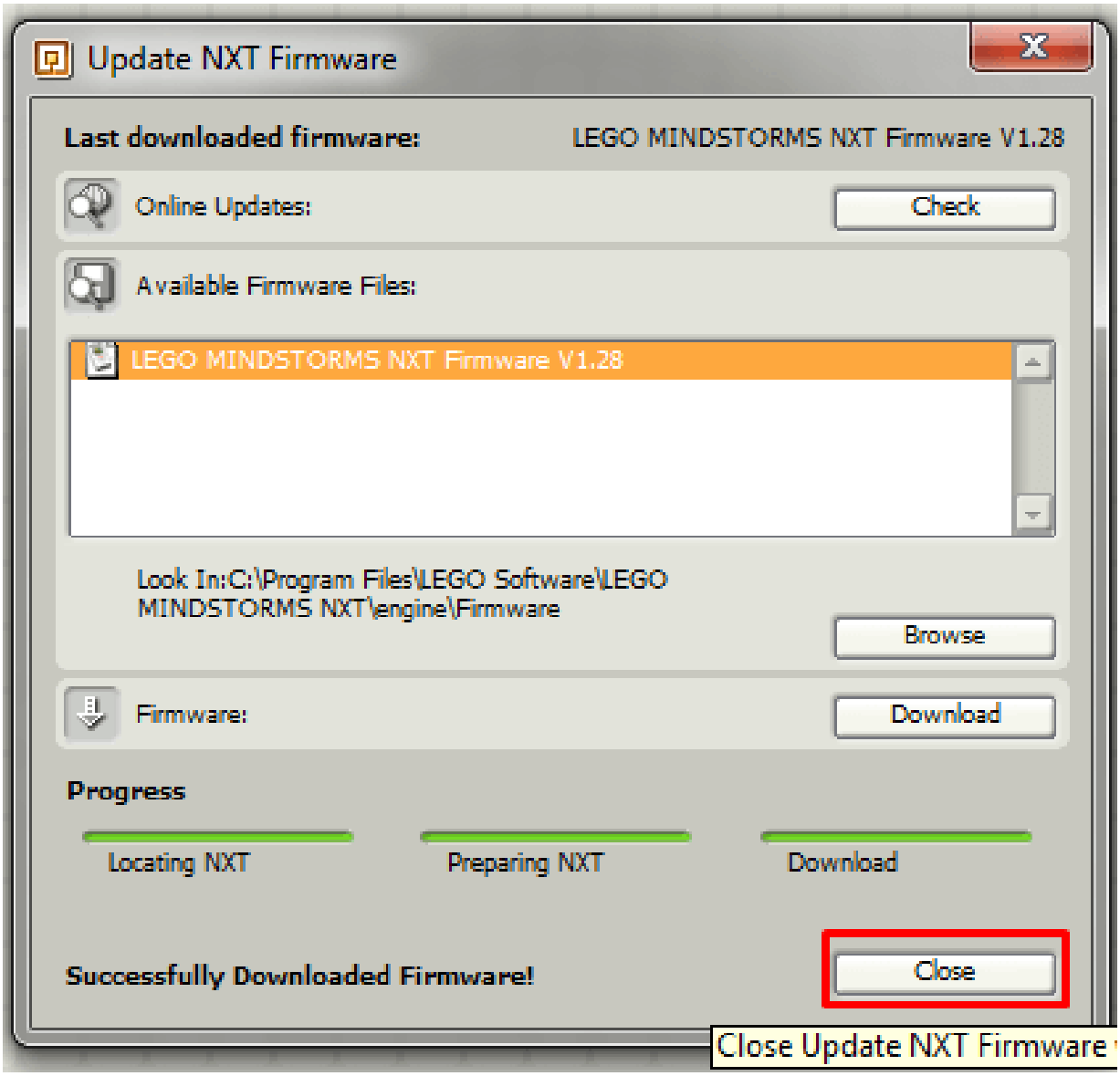



Figura 5.4: Realizando o update - passo 3

Agora basta ligar o NXT e conectá-lo ao computador. Feito isso, é só abrir o software do EV3, seja a versão Home Edition ou a Education. No canto inferior direito do software pode-se ver se a conexão foi realizada com sucesso; caso tenha sido, já se pode programar e fazer download dos programas para o NXT.

**Observação**

Apesar de o software não apontar incompatibilidade, o bloco *Colour Sensor* não serve para controlar nem o sensor de cor, nem o sensor de luminosidade do kit NXT. Entretanto, isso não inviabiliza a utilização deles. Ao final deste livro será mostrado como criar um programa seguidor de linha (que faz uso do sensor de luminosidade) para o NXT 2.0.

# Capítulo 6
# Tools

Nessa aba há ferramentas que podem ser usadas para auxiliar na construção do programa, constituindo elementos essenciais à criação de projetos. A aba *Tools* contém ferramentas muitos úteis na construção dos projetos. Ela possui também um recurso vital que já vimos: o *Firmware Update*, com o qual é checado se o Firmware do Brick está atualizado; caso não esteja, é possível atualizá-lo. Dentre as ferramentas encontradas aqui, pode se destacar também, o *Sound Editor* e o *Image Editor*, utilizados para a edição de som e imagem respectivamente. Todos esses recursos já citados e os outros presentes na aba *Tools* serão abordados neste capítulo.

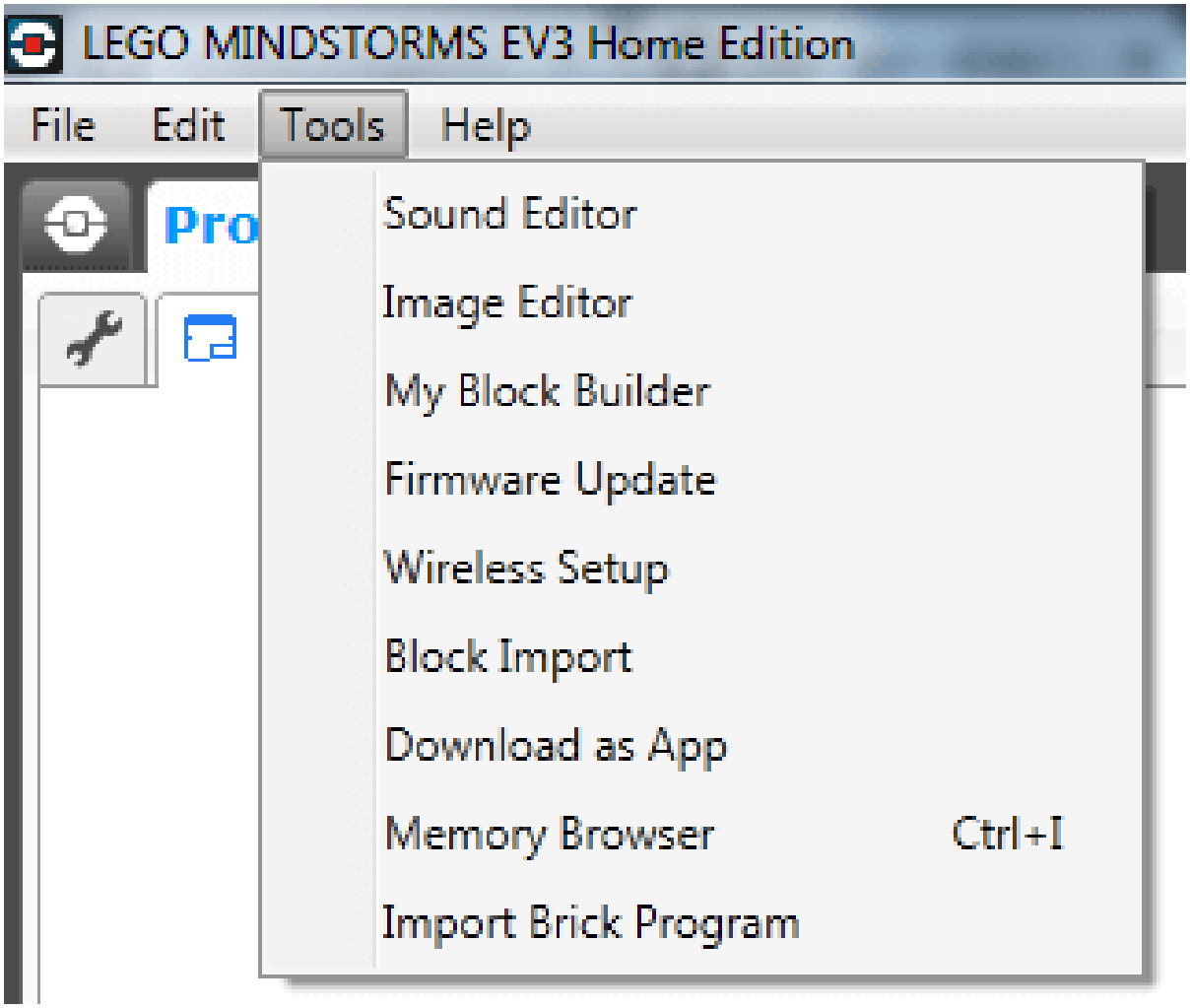



Figura 6.1: Aba Tools

**Sound Editor**

Com esta ferramenta, pode-se carregar um arquivo de som, ou então utilizar o microfone do computador para gravar um áudio. O editor suporta aproximadamente 8 segundos de um arquivo.

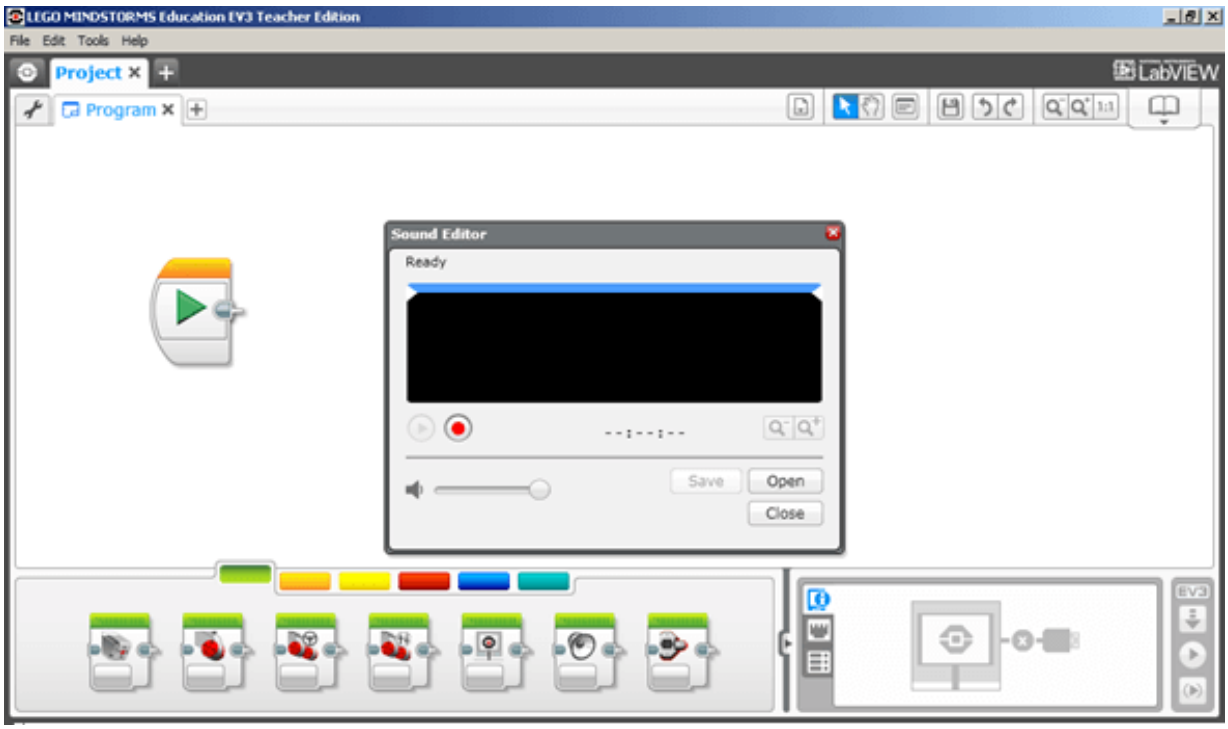



Figura 6.2: Sound Editor

Para gravar, clica-se em `Record` (bolinha vermelha). Para carregar um arquivo, pressiona-se o botão `Open` . Os formatos suportados são: `rso` , `rsf` , `wav` e `mp3` .

## Image Editor

Com esta ferramenta, pode-se editar uma imagem para utilizar no programa e exibir no display. É possível desenhar ou escrever algo no editor, além de carregar uma imagem, cujas dimensões são ajustadas automaticamente. O Image Editor aceita os formatos: `jpg` , `jpeg` , `rgf` e `png` .

A área de desenho é muito semelhante ao famoso “Paint”, porém é mais básica e conta apenas com algumas ferramentas, mais que suficientes para cumprir seu propósito.

Para carregar uma imagem, clica-se em `Open` . Para abrir um novo editor, pressiona-se `New` e para salvar, marca-se a aba `Save` . No EV3 Preview que fica situado no canto direito, é possível acompanhar o

resultado prévio do trabalho. No canto direito, estão as ferramentas que podem ser utilizadas para a criação da imagem.

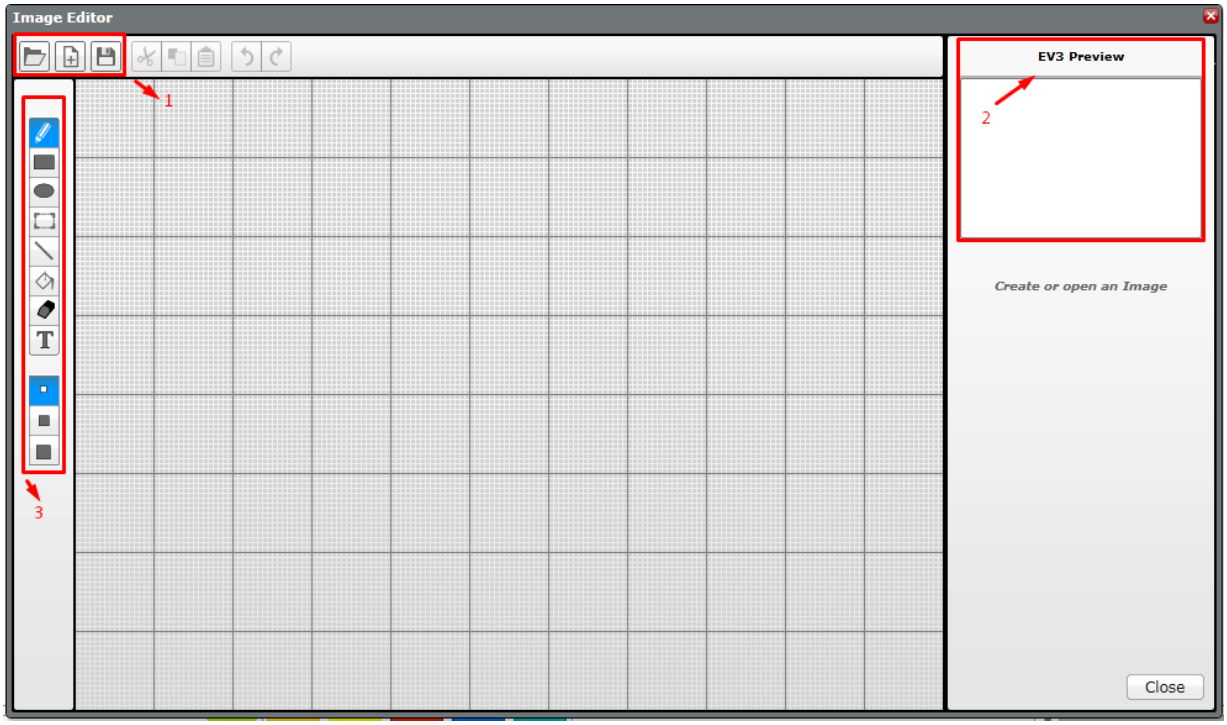



Figura 6.3: Image Editor

1. `Open` , `New` , `Save` .
2. EV3 Preview
3. Ferramentas para edição

## My Block Builder

Muitas vezes, na construção de um programa, é necessário utilizar uma sequência de blocos mais de uma vez. Isso significa que seria necessário repetir toda a mesma sequência. O *My Block Builder* foi criado para solucionar este problema. Com ele, é possível transformar uma sequência de blocos em um só bloco. E depois de criado, pode-se utilizá-lo inúmeras vezes. Isso poupa tempo e deixa o programa muito mais organizado, sem dúvida, é um recurso muito útil e importante. Por isso, sua função e funcionamento serão abordados com mais detalhes em uma seção especial.

## Firmware Update

É utilizado para atualizar o Brick com a versão mais recente do firmware. Ao conectar o Brick pela primeira vez a atualização pode ser pedida automaticamente, para que o Brick esteja preparado para uso. Com o Brick conectado à máquina é só clicar em `Update Firmware` .

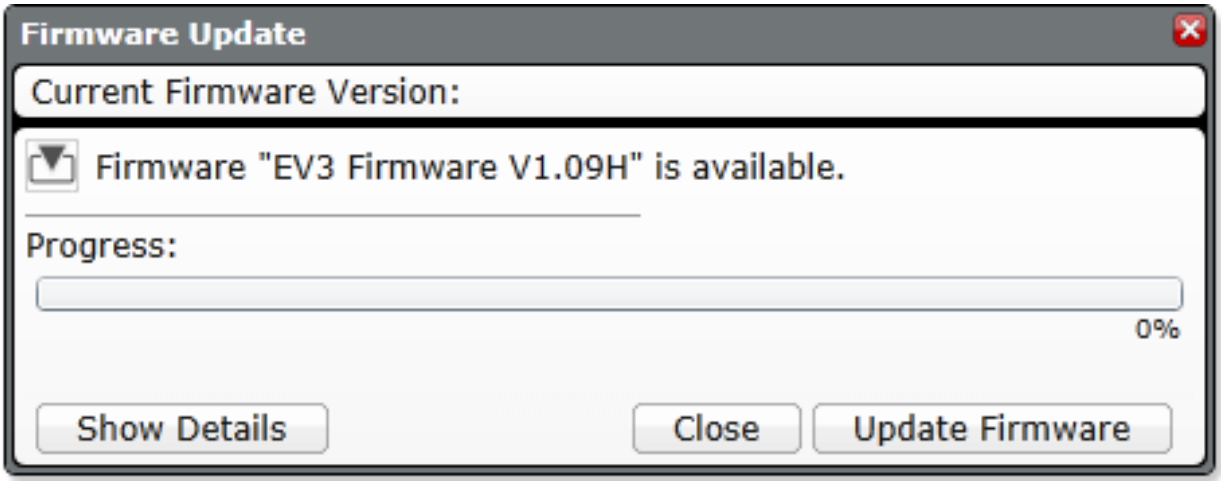



Figura 6.4: Firmware Update

No botão `Show Details` é possível ver os detalhes da atualização (caso disponível).

## Wireless Setup

O Brick conta com um recurso que permite a conexão Wi-Fi. Contudo, para que a conexão seja realizada, é preciso acoplar um dispositivo externo à porta USB (situada na lateral) do Brick, um Dongle Wi-Fi. O único Dongle Wi-Fi oficialmente suportado pelo Brick EV3 é o **Netgear N150 Wireless Adapter (WNA1100)**. A conexão Wi-Fi permite que seja dispensado o uso do cabo, desta forma é possível controlar o Brick remotamente. A ferramenta *Wireless Setup* é usada para configurações Wireless e permite ajustar o Brick para se conectar a uma rede sem fio.

O Dongle (até o momento em que este livro foi escrito) não foi encontrado em nenhuma loja do Brasil. Pela dificuldade de acesso no país, ele não será abordado com maior detalhamento e profundidade.

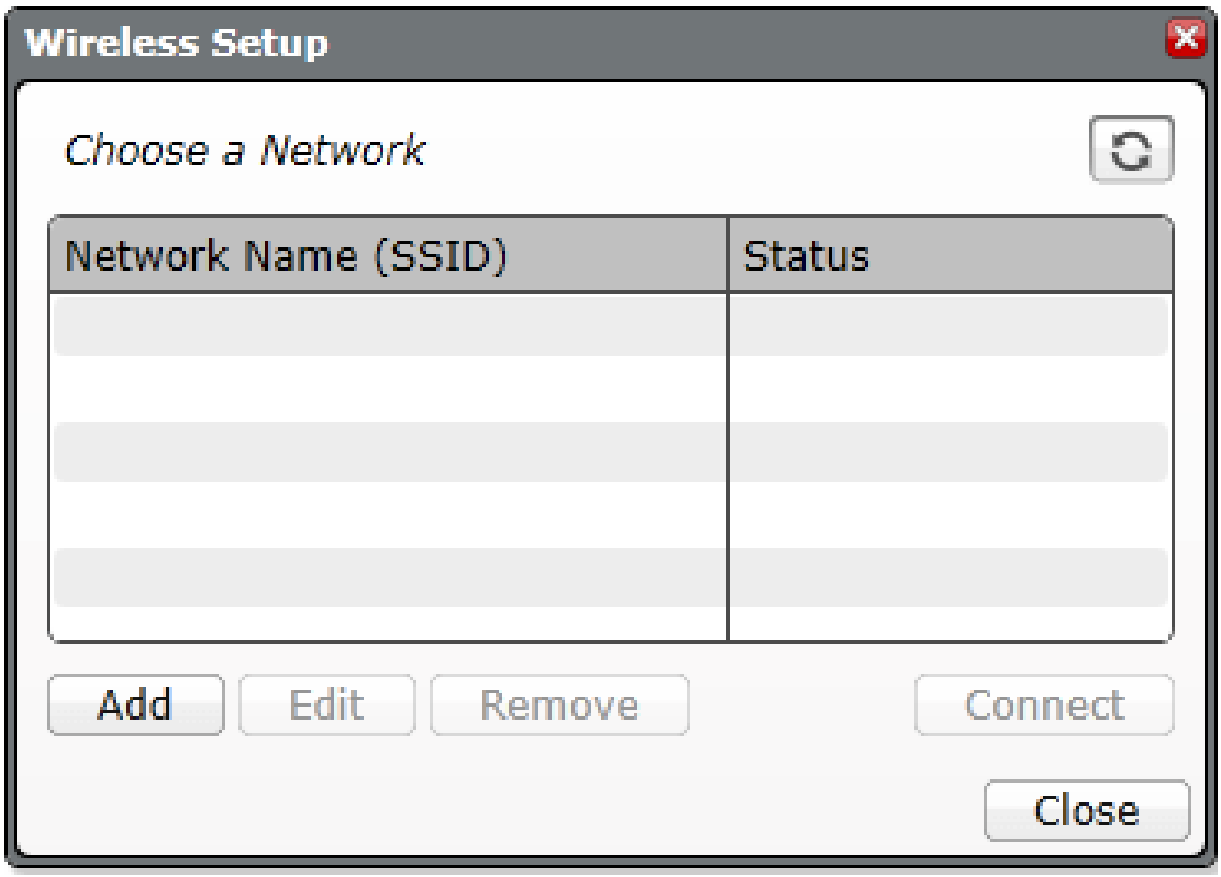



Figura 6.5: Wireless Setup

## Download as App

Esta ferramenta permite que você faça o download de um programa como um App para o seu Brick. Após feito o download, basta ir em *Brick Apps* e procurar o App pelo nome. Navegando pelos *Brick Apps*, você encontra alguns aplicativos previamente instalados que você pode utilizar para tarefas como controle do motor e visualização das portas, por exemplo. Com a ferramenta, os seus programas podem ser baixados como Apps e acessados

diretamente no *Brick Apps*. Sendo assim, você pode criar os seus próprios Apps.

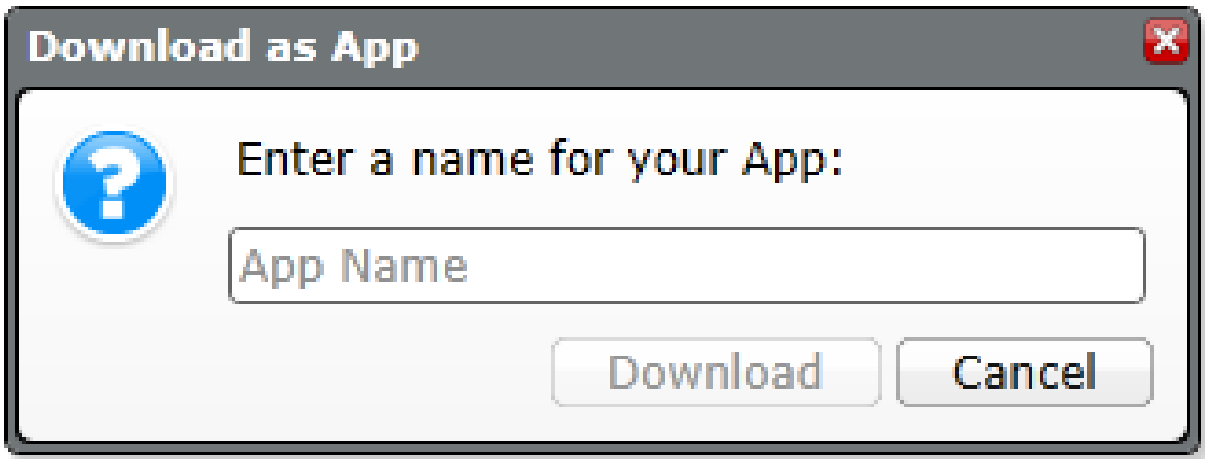



Figura 6.6: Download as App

### Block Import and Export Wizard

Utiliza-se essa ferramenta para importar um bloco que foi baixado. É possível baixar blocos adicionais para o software. Por exemplo, quem utiliza a Home Edition pode baixar um bloco necessário para o seu programa. Após feito o download, essa ferramenta é utilizada para importá-lo ao software.

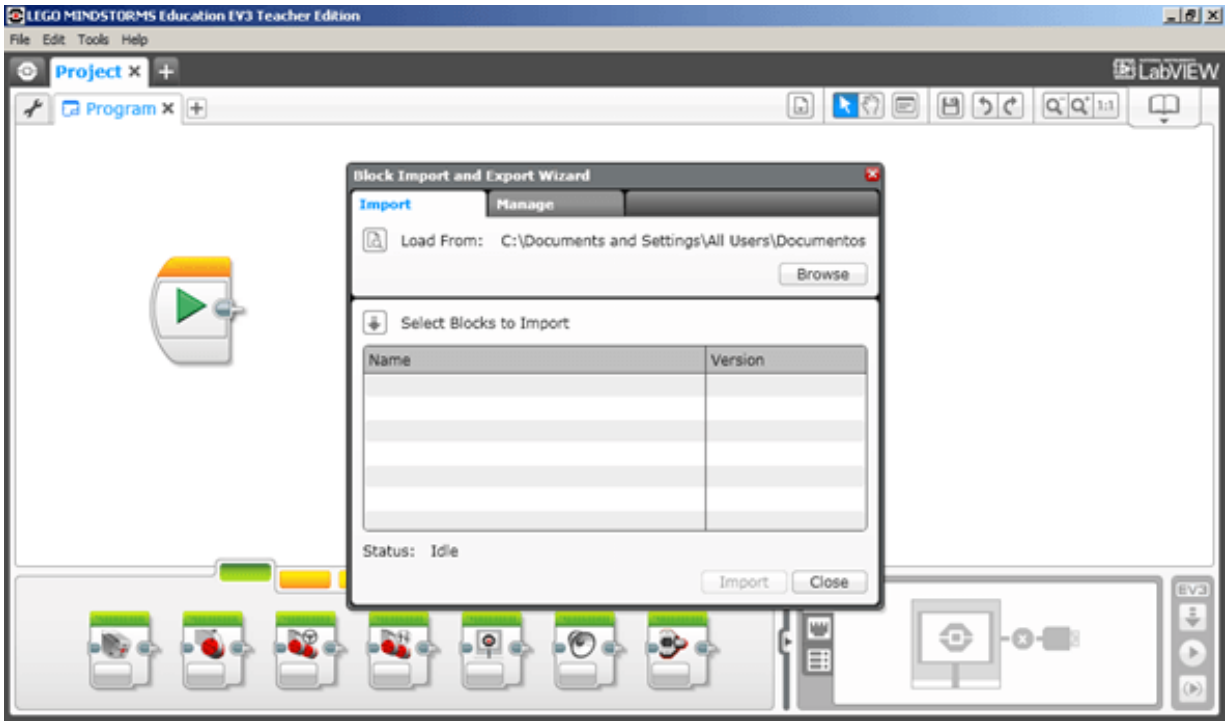



Figura 6.7: Block Import and Export Wizard

## Importando blocos para o EV3 Home Edition

A versão Home Edition do software contém somente os blocos sensores que acompanham o kit. Mas isso não impede a utilização de outros sensores para o robô. É possível baixar blocos sensores adicionais e instalá-los no software, permitindo sua utilização.

Figura 6.8: Paleta de blocos

## Fazendo download do bloco

O site da LEGO® disponibiliza o download de blocos adicionais pelo seguinte link:

https://www.lego.com/en-us/mindstorms/downloads

Os blocos são os seguintes:

- *Energy Meter Block*
- *Gyro Sensor Block*
- *Sound Sensor Block*
- *Temperature Sensor Block*
- *Ultrasonic Sensor Block*

Cada um desses sensores será abordado na parte três do livro, suas funções e modo de funcionamento, para você saber como decidir qual é o mais adequado para seus futuros projetos.

Ao escolher, basta clicar no bloco desejado, que o download será feito automaticamente.

EV3 SOFTWARE BLOCK DOWNLOAD (PC/MAC)
Missing a block in the software? Here you can download blocks for the extra sensors.
EV3 SOFTWARE BLOCKS
Download these software updates to activate each of these additional sensors in your EV3 Programmer software (PC/MAC only).
Energy Meter Block
Gyro Sensor Block
Sound Sensor Block
Temperature Sensor Block
Ultrasonic Sensor Block

Figura 6.9: Fazendo download do bloco

## Instalando o bloco

Após feito download do bloco é necessário instalá-lo no software. Para isso, utiliza-se a ferramenta `Block Import` , presente na aba *Tools*.

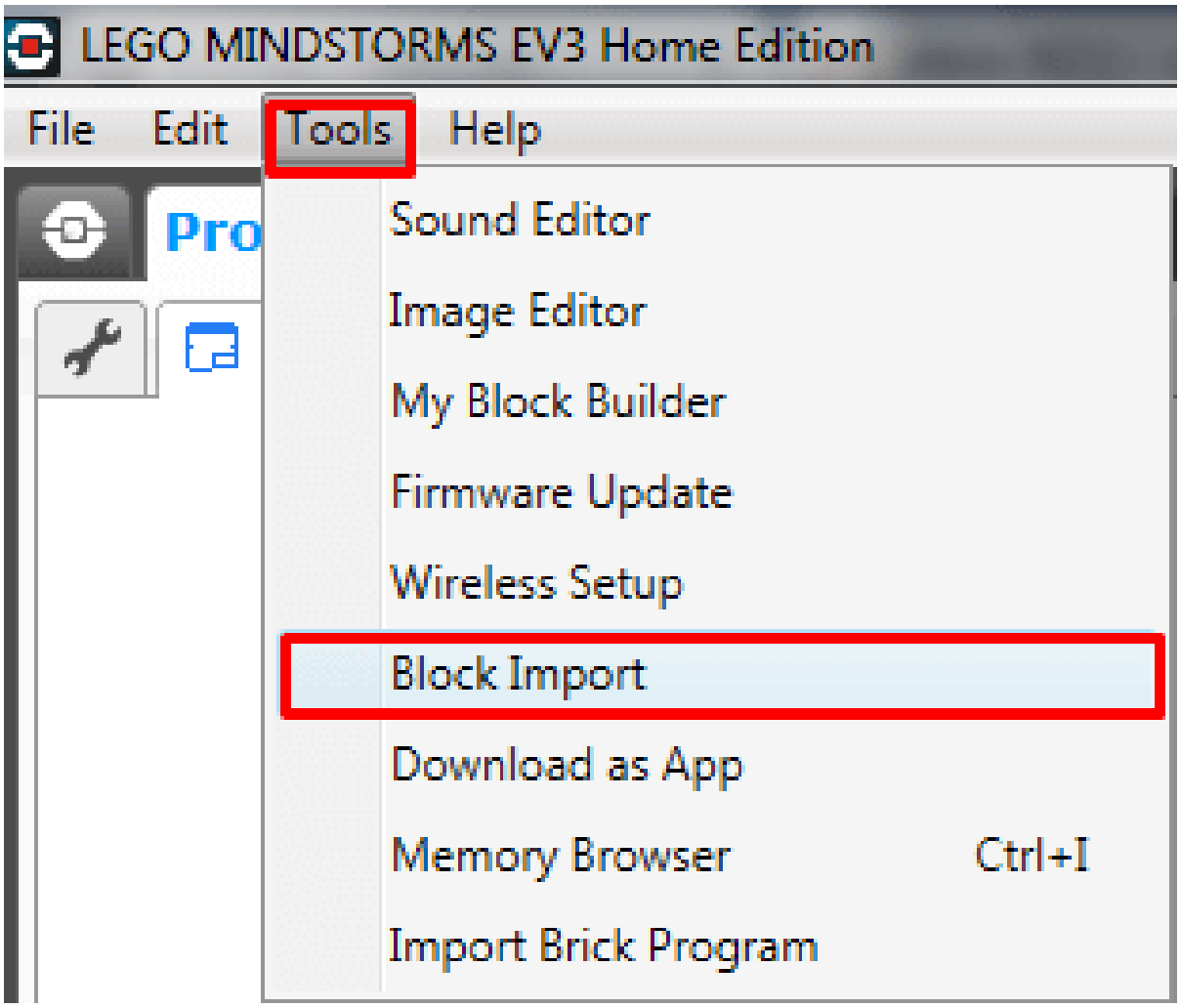



Figura 6.10: Instalando o bloco - passo 1

Após clicar em `Block Import` a seguinte tela aparecerá:

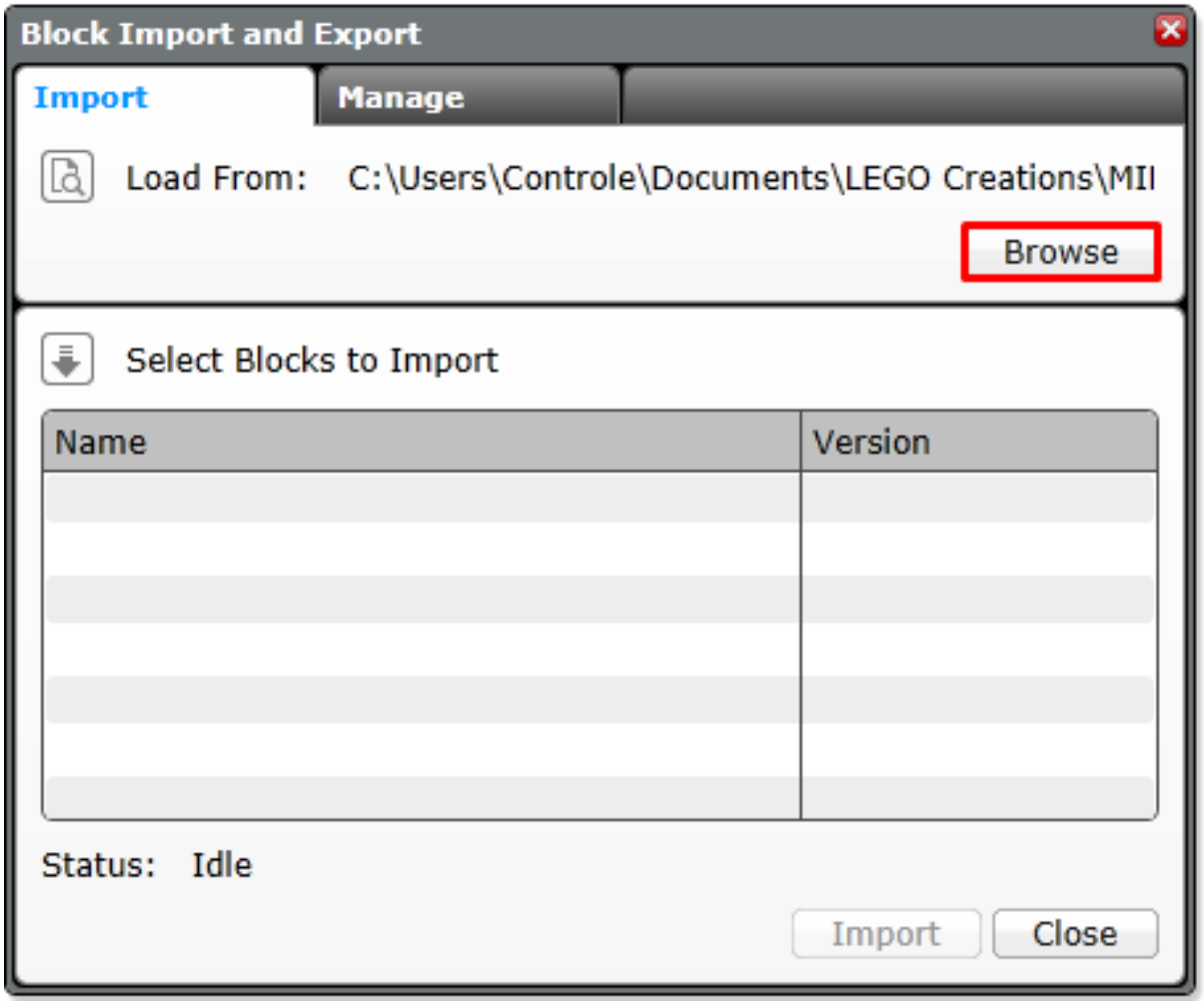



Figura 6.11: Instalando o bloco - passo 2

O próximo passo consiste em pressionar o botão `Browse` para selecionar o arquivo, cuja pasta padronizada é `Download` . Caso o arquivo esteja em outra pasta, é preciso localizá-la antes de selecioná-lo. Em seguida, deve-se clicar no arquivo e depois em `Abrir` , ou simplesmente dê um clique duplo.

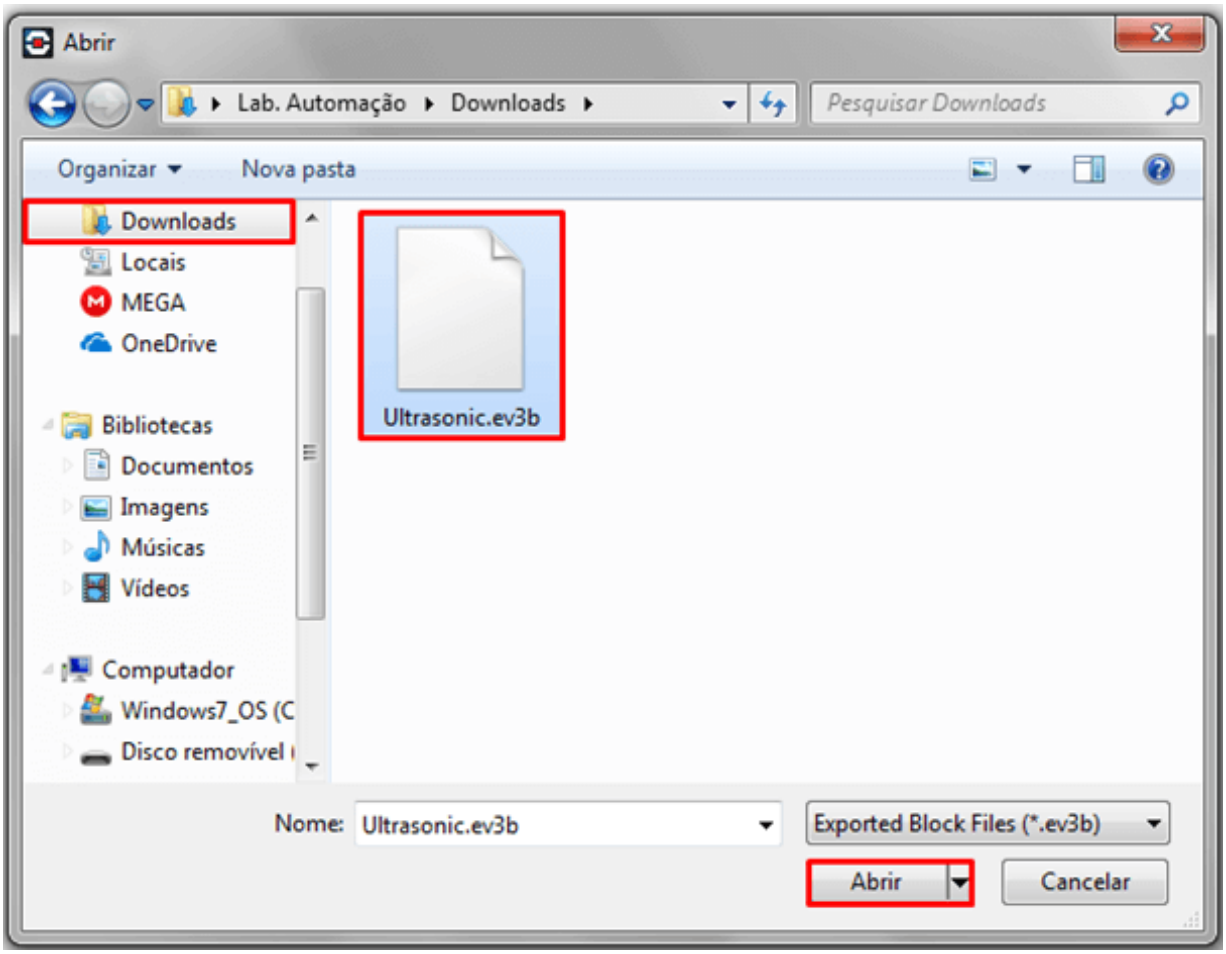



Figura 6.12: Instalando o bloco - passo 3

Em seguida é só clicar em `Import` .

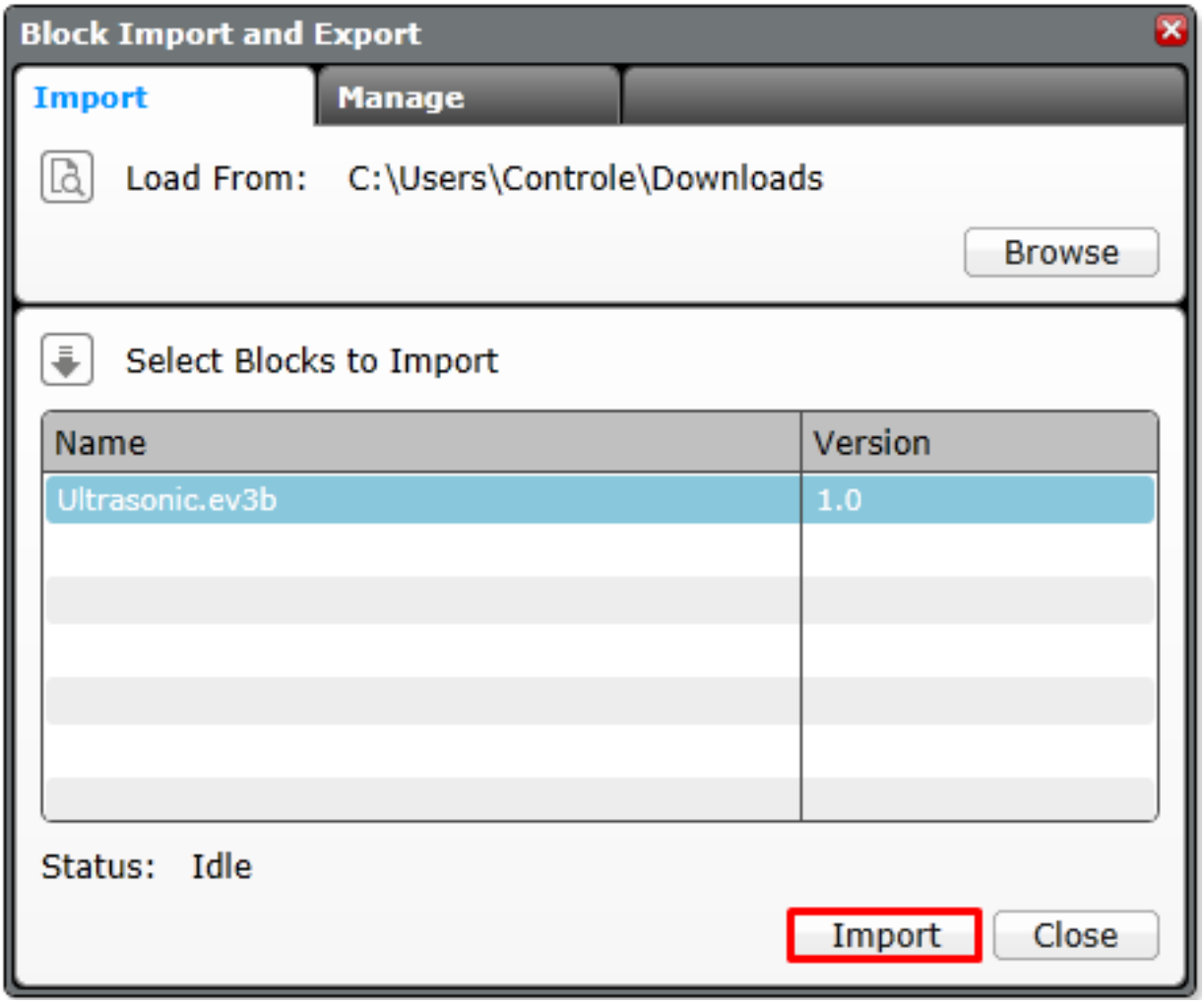



Figura 6.13: Instalando o bloco - passo 4

Para que a importação do bloco tenha efeito, deve-se reiniciar o programa, conforme a mensagem que aparecerá após clicar em `Import` . Basta pressionar o botão `Ok` , que, em seguida, fecha-se o programa, para reiniciá-lo novamente

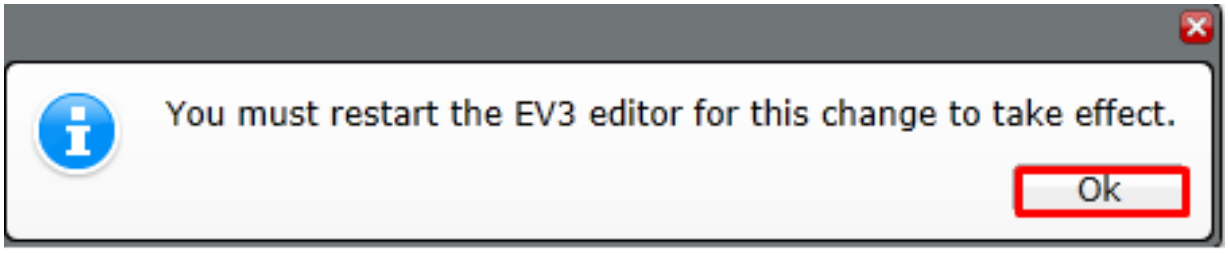



Figura 6.14: Instalando o bloco - passo 5

Agora o programa já conta com o novo bloco.

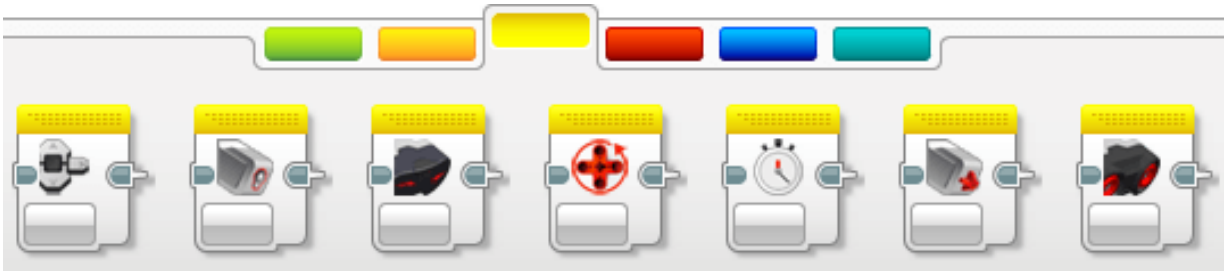


Figura 6.15: Instalando o bloco - bloco instalado

## Memory Browser

Aqui pode ser feito o gerenciamento da memória do Brick, inclusive no cartão SD, caso ele esteja disponível.

É possível ver o espaço livre e o espaço ocupado pelos projetos e aplicações. Pode-se copiar arquivos, apagar, fazer upload. Para isso, basta o Brick estar conectado à máquina.

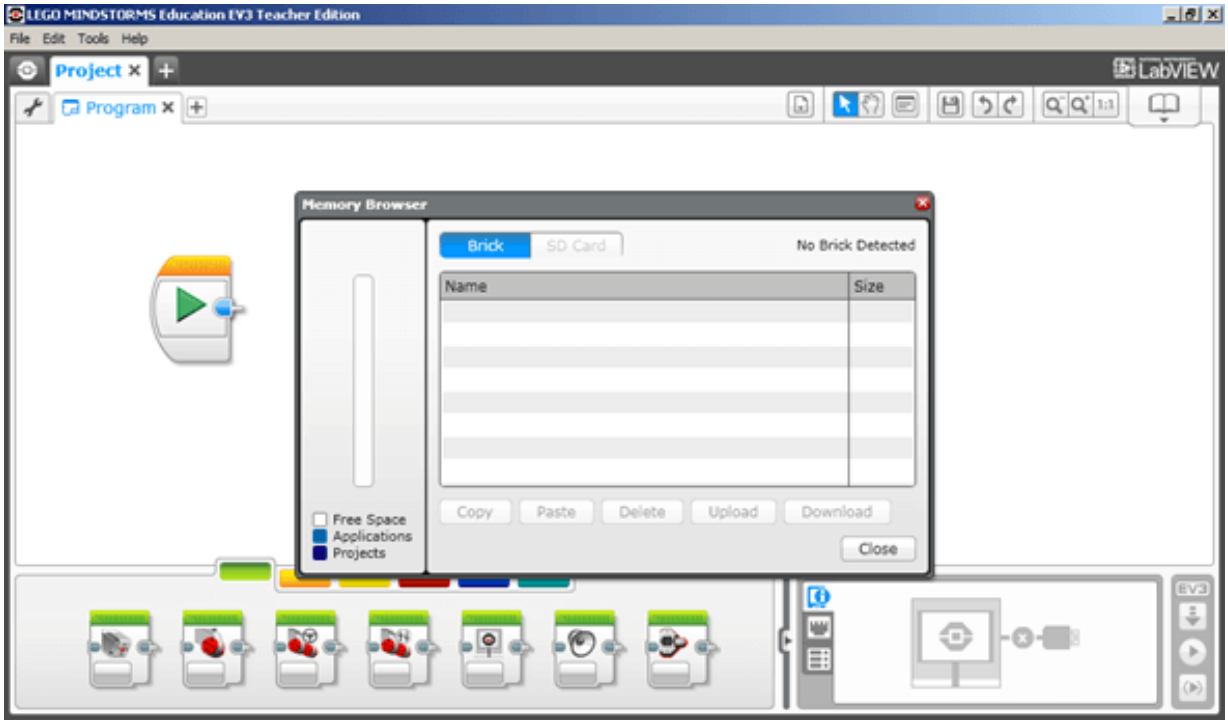



Figura 6.16: Memory Browser - 1

Com o Brick conectado, visualiza-se o nome dos arquivos e o espaço ocupado por eles em sua memória.

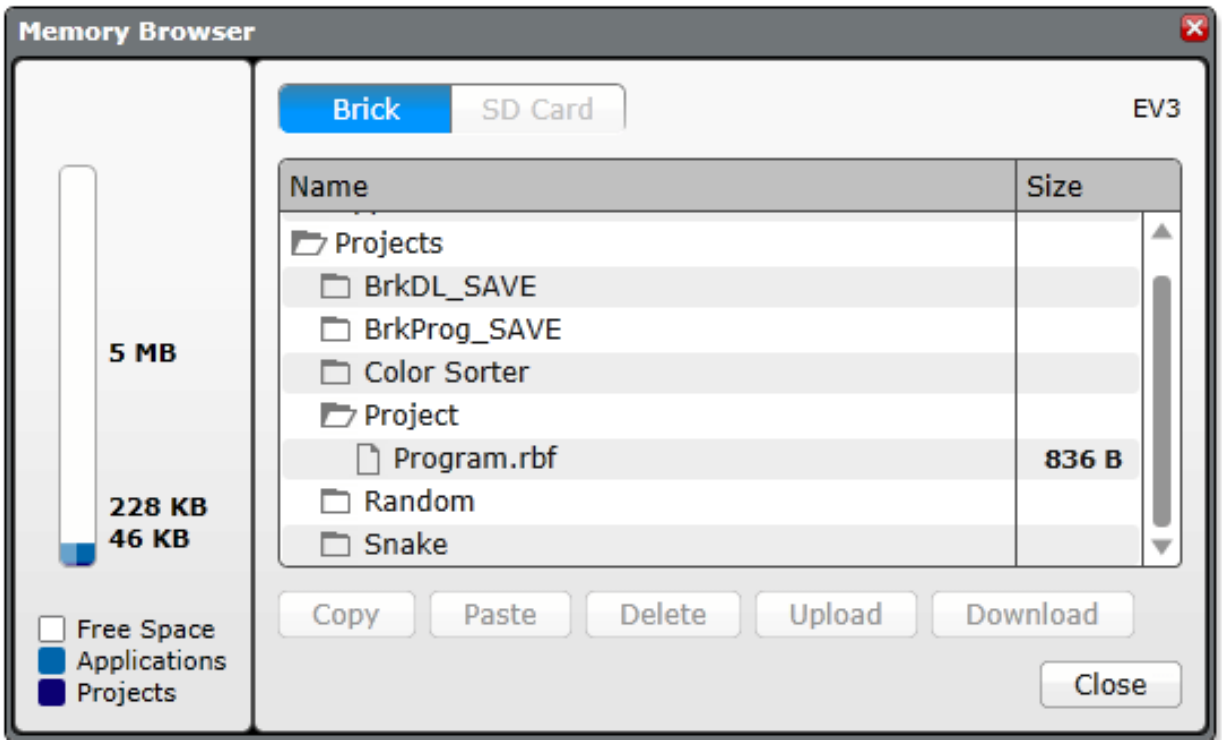



Figura 6.17: Memory Browser - 2

Para exemplificar todas informações, a seguir, será mostrado como fazer o download de um programa para o Brick. Depois de efetuado o download, as propriedades do *Memory Browser* serão mostradas.

Observe o seguinte projeto, trata-se de um robô que segue uma linha preta no chão. Mas, não se preocupe em entender o programa agora, mais à frente ele será visto.

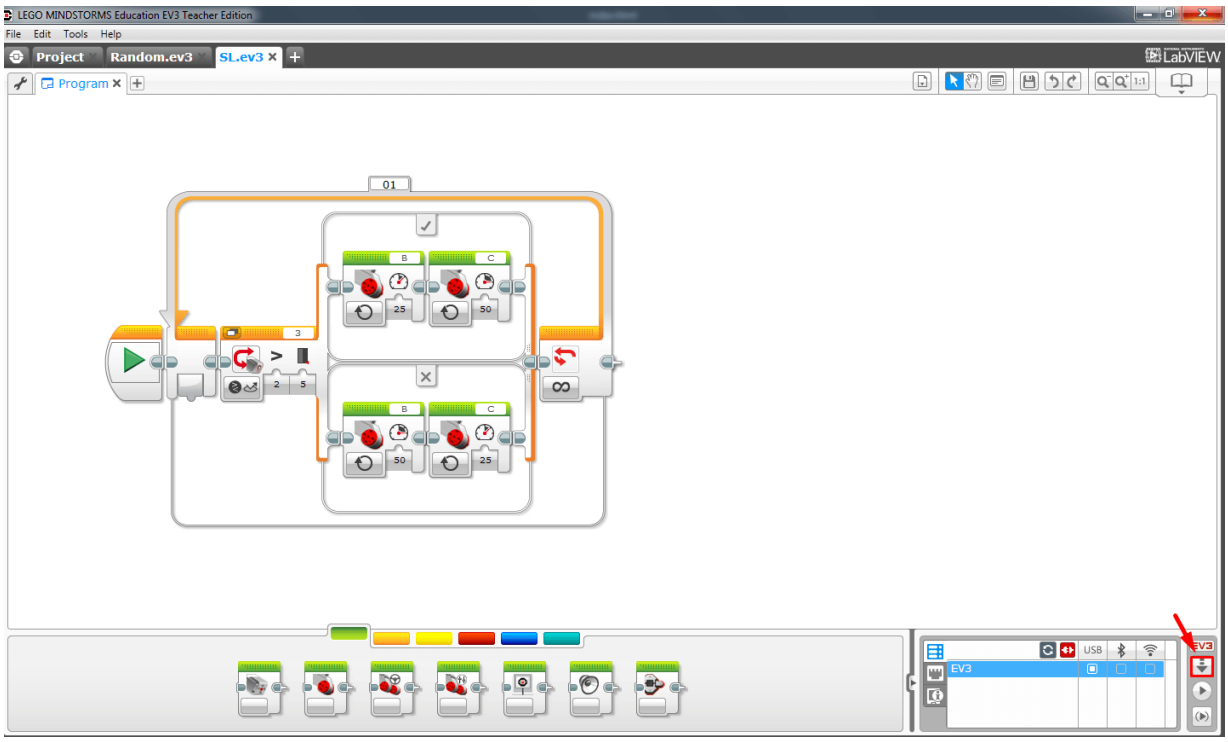



Figura 6.18: Robô que segue linha

No canto inferior direito, está a *Hardware Page*. A seta está apontada para o botão `Download` , responsável por carregar o programa no Brick. A próxima imagem mostra a *Hardware Page* com mais detalhes.

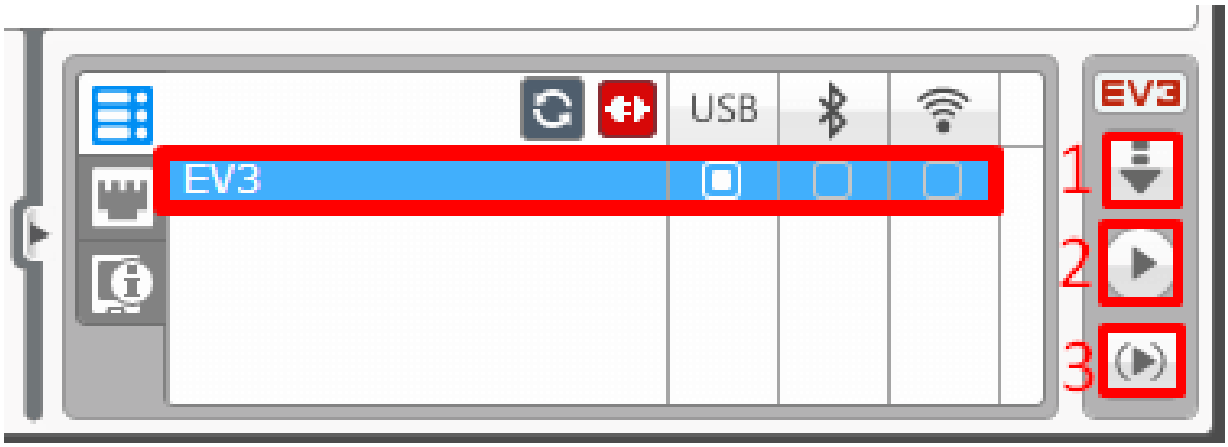



Figura 6.19: Hardware Page: 1 - Download, 2 - Download and Run, 3 - Run Selected

Observe o retângulo, dentro dele está escrito "EV3" e a opção "USB" está marcada. Isto indica que a conexão entre o Brick e o EV3 está sendo realizada por meio do cabo USB. Sempre ao conectar o Brick esta tela deverá aparecer.

> **Observação**
>
> É necessário que o Brick esteja ligado para que a conexão seja mostrada. Caso seu Brick esteja conectado ao PC e desligado, não será possível efetuar conexão.

O primeiro botão `Download` é usado para transferir o programa para o Brick. Dessa forma, é necessário selecionar a pasta do projeto no Brick e executar o programa. A segunda opção disponível é o `Download and Run` , com ela, o programa é executado logo após o download ser efetuado. Por último, temos o botão `Run Selected` , com o qual é possível selecionar apenas a parte desejada do programa e executá-la.

Agora as funções do *Memory Browser* podem ser mais facilmente compreendidas. Conforme já foi visto, o *Memory Browser* pode ser acessado na aba *Tools*, mas há outra forma de fazê-lo também. A imagem a seguir ilustra isso.

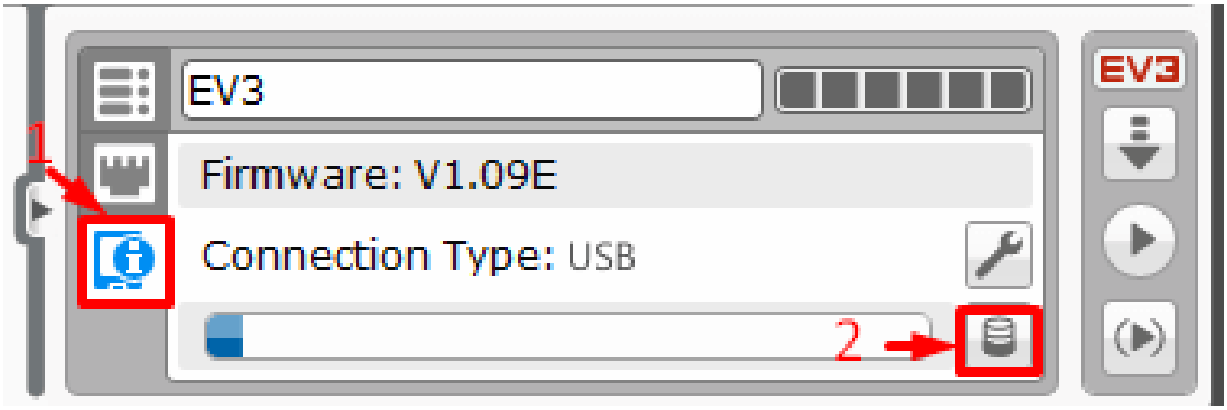



Figura 6.20: Acessando o Memory Browser pela Hardware Page

O número "1" indica o primeiro passo, que é clicar no botão `Brick Information`. O segundo passo é clicar em `Open Memory Browser`.

Feito isso, é possível explorar a memória do Brick, realizando as opções que o *Memory Browser* fornece. Como foi feito o download para o Brick, a pasta do projeto (que foi nomeado por "SL") deverá estar contida na pasta `Projects`.

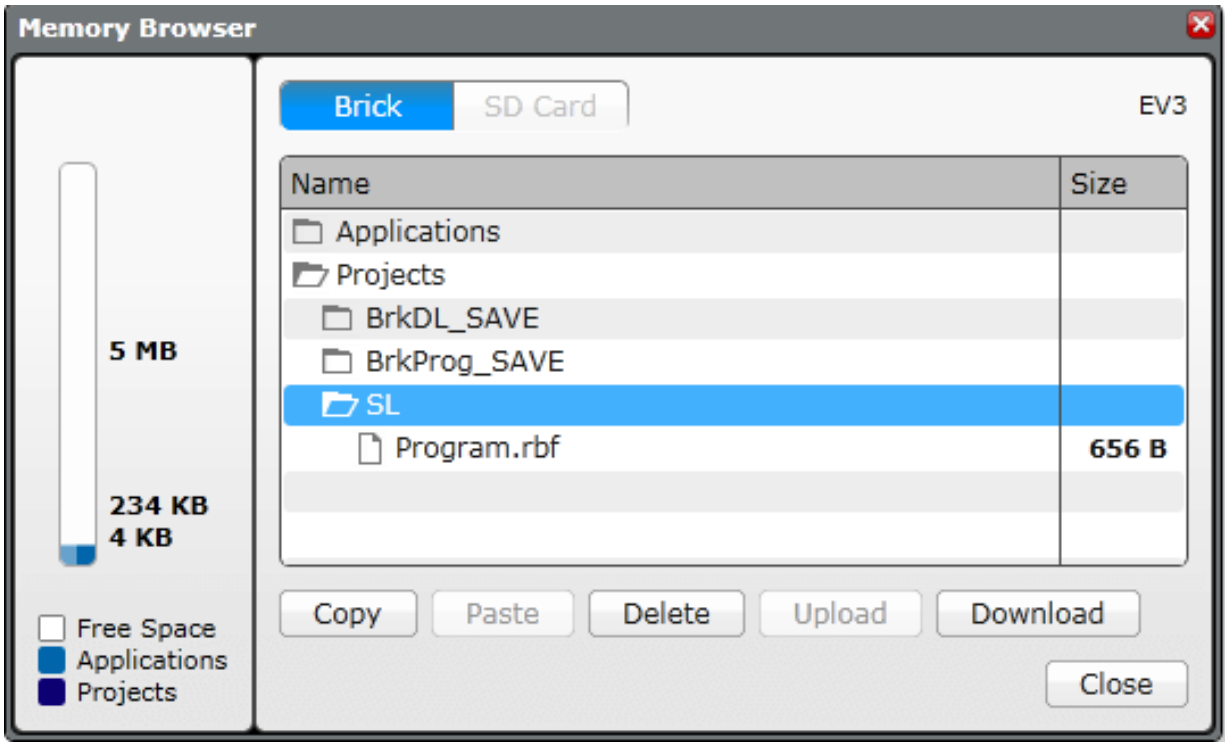



Figura 6.21: Acessando o Memory Browser - Explorando a memória do Brick

No canto esquerdo, está uma coluna que representa a memória do Brick. A legenda contém as seguintes informações:

- *Free Space* - designa o espaço livre disponível no Brick;
- *Applications* - o espaço ocupado pelos Apps (eles já vêm no Brick);
- *Projects* - o espaço ocupado pelos projetos.

Como o leitor percebeu, no Brick, apenas 5 MB de memória estão disponíveis. Isso pode parecer muito pouco, mas é o suficiente para

o download de muitos projetos. Como é possível ver, o projeto que foi baixado para o Brick contém apenas 590 B de informação.

**Outras ferramentas disponíveis na versão Education**

O leitor e usuário da versão Education deve ter percebido que em sua aba *Tools* havia mais recursos disponíveis. Eles são os seguintes:

- *Data Log File Manager*
- *Remove Values from Dataset*
- *Create Data Logging Program*
- *Export Datsets*

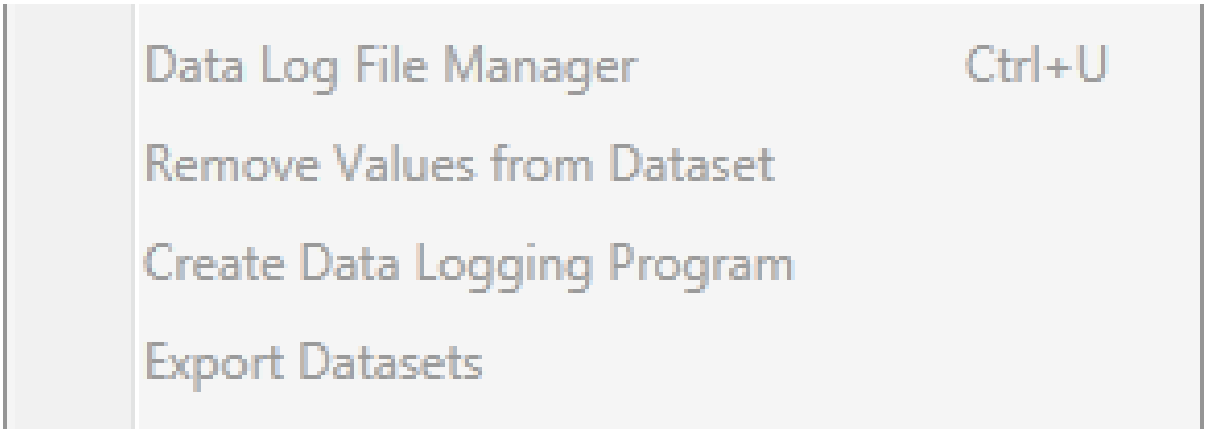



Figura 6.22: Ferramentas para Experimentos

Estas ferramentas são usadas em **Experimentos** (*Experiment*). Contudo, elas não serão abordadas neste livro pelos seguintes motivos:

- É recurso mais avançado, e pode atrapalhar o leitor que ainda não teve contato com programação, pois certos conhecimentos prévios são necessários;
- Tem um enfoque mais científico, ideal para aplicações em aula, que não é o objetivo em si do livro.

- Muitas páginas seriam necessárias para isso, por consequência, acabaria estendendo demais o material.

**EXPERIMENTOS**

O recurso *Experiment* está presente apenas na versão Education. Ao criar um projeto, você pode optar entre criar um programa (conforme já foi visto) e um experimento. O experimento permite monitorar e analisar os dados coletados pelos sensores acoplados ao Brick.

# Capítulo 7
# Conclusão

Nesta parte do livro, você viu desde a instalação até os primeiros passos no software. Esse contato e conhecimento com o software e suas ferramentas é indispensável para a construção de bons projetos. Para programar bem, antes de tudo, é necessário conhecer as possibilidades que o ambiente de programação oferece.

Na próxima parte do livro será iniciado o contato com a programação. Ela tratará dos blocos oferecidos pelo software. Veremos como utilizar o bloco, como configurá-lo, e claro, sua função. Ao final da parte 3 você já terá uma breve noção de como construir seus próprios programas.

# A LINGUAGEM EV3-G

Em uma linguagem de programação alguns conceitos são amplamente usados e são de suma importância, pois constituem a base da linguagem. Esta parte do livro, apesar de curta, é fundamental. Os capítulos aqui dispostos lhe oferecerão a base necessária para que você tenha uma maior facilidade no aprendizado. Após estudá-los, você já terá uma base em programação, mesmo não tendo desenvolvido nenhum programa ainda. Com estes capítulos você conseguirá se situar melhor e obterá um melhor entendimento dos blocos e suas funções. Quando chegar aos primeiros programas, você será capaz de presumir e, até mesmo, já saber alguns dos passos necessários para a solução da atividade proposta.

## Capítulo 8
## Conhecendo o ambiente de programação

### Vamos conhecer um pouco mais sobre os blocos de programação?

Figura 8.1: Bloco Start

Na próxima imagem, está a indispensável **Paleta de Programação**. Nela, encontram-se todos os blocos necessários à construção do nosso programa. Ela contém divisões; os blocos que se encontram nela são organizados por categoria, para facilitar a programação. A seguir será feita uma abordagem mais profunda da Paleta de Programação.

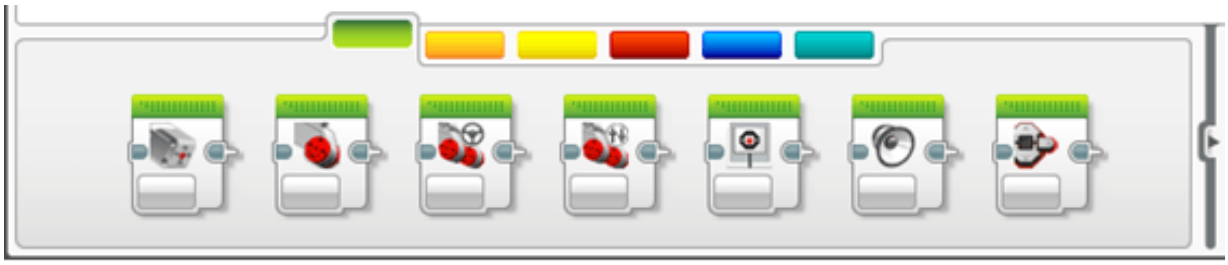


Figura 8.2: Paleta de Programação

No canto inferior direito está a *Hardware Page*, que já conhecemos. Ela é responsável por gerenciar a comunicação do Brick, além de mostrar a atuação de periféricos de entrada e saída conectados ao Brick. Na *Hardware Page* é realizado o download do código para o Brick.

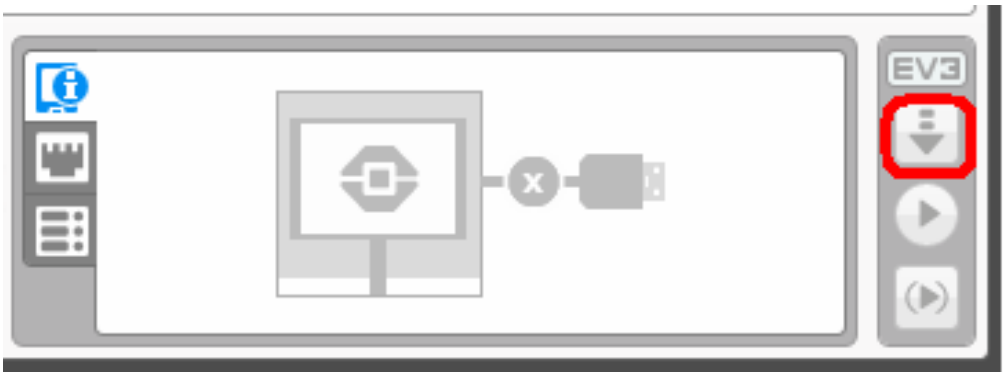



Figura 8.3: Hardware Page

Todos esses recursos serão abordados de forma clara, detalhada e concisa nos próximos capítulos desta parte.

# Capítulo 9
# Blocos de programação

Para criar um programa no EV3, basta pensar em uma sequência de passos bem definidos, e fazê-los por meio de blocos. Esse processo pode ser um simples programa com poucos blocos, ou até mesmo algo de maior complexidade que envolva diversos tipos de blocos. Imagine o quão trabalhoso seria ter de procurar cada um desses blocos sem qualquer tipo de orientação ou categoria.

Para facilitar o trabalho do programador, ou seja, o seu trabalho, os blocos se encontram dispostos em categorias, conforme suas funções, que podem ser: movimento, controle, sensores, dados, blocos avançados, e ainda, a opção *My Blocks*, (utilizada para criar um só bloco com uma sequência de bloco já existente, permitindo reutilizá-la em diversas partes do código).

## 9.1 Blocos de ação (Action Blocks)

São blocos utilizados para o controle de ações do programa. Com eles, pode-se controlar Display (tela exibida), Status light (luz de status), emissão de sons pelo Brick, além dos motores propriamente ditos.

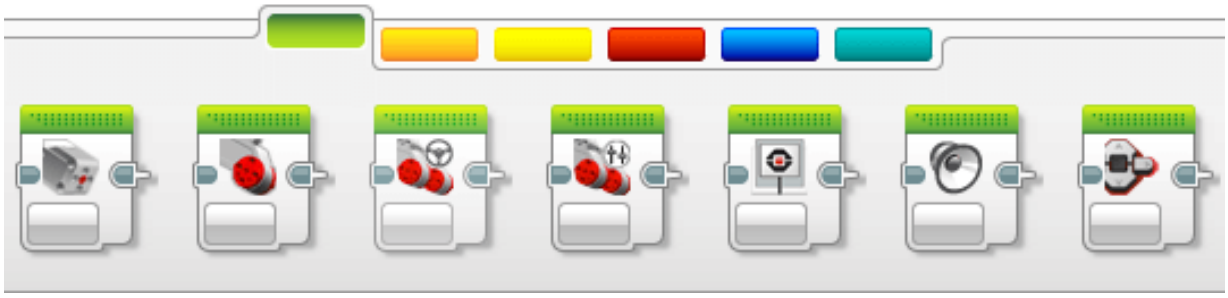


Figura 9.1: Blocos de ação (Action Blocks)

## 9.2 Blocos de controle (Flow Blocks)

Estes blocos são estruturas de controle do programa. Eles são cruciais e importantíssimos na construção de um programa. O primeiro bloco dessa paleta é `Start` , com isso, você já deve ter percebido o quão grande é a importância dela. Podemos dizer que esses blocos dão vida ao programa, e o exemplo do bloco `Start` demonstra isso perfeitamente.

Controlar o programa é dar dinamismo a ele: quando ele deve iniciar, se ele deve esperar (por tempo, ou evento, por exemplo) para prosseguir as ações, quantas vezes ele deve repetir as sequências de blocos, quando parar de executá-los e, ainda, ter poder de decisão. Por exemplo, nesta paleta estão contidos os blocos `Switch` e `Loop` (que trabalharemos posteriormente). Esses blocos encapsulam outros blocos. Dessa maneira você controla todos os blocos neles contidos.

Para criar um programa, deve-se começar com o bloco `Start` . Pode-se ainda configurar alguns desses blocos de acordo com o modo de operação desejado.

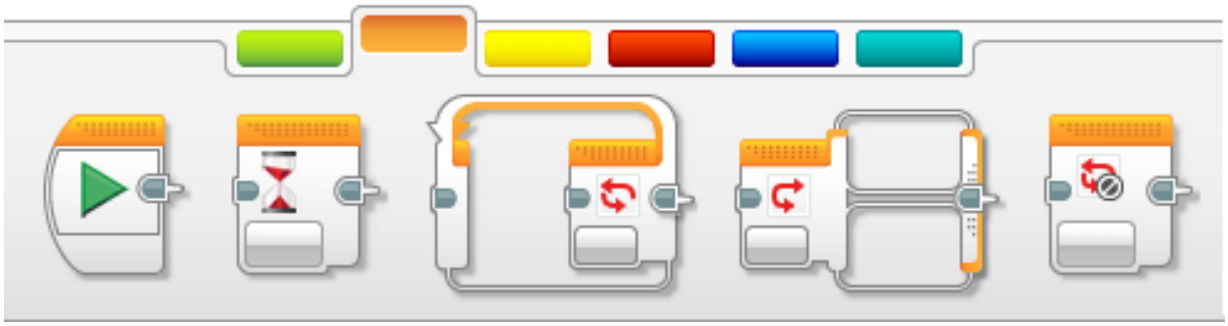


Figura 9.2: Blocos de controle (Flow Blocks)

## 9.3 Blocos de sensores (Sensor Blocks)

Com estes blocos, é possível controlar os sensores conectados à entrada do robô, por meio dos quais se obtêm as informações do mundo exterior, como sensores táteis (de toque), ultrassônicos, detectores de cor, distância etc.

Figura 9.3: Blocos de sensores (Sensor Blocks)

## 9.4 Blocos de dados (Data Blocks)

Com estes blocos, podem-se realizar diversas operações matemáticas. Eles são responsáveis pela lógica, cálculos matemáticos e armazenamento de dados, através de variáveis ou constantes.

Figura 9.4: Blocos de dados (Data Blocks)

## 9.5 Blocos avançados (Advanced Blocks)

Esta categoria possui blocos utilizados em programas mais complexos; com eles, pode-se configurar conexão via bluetooth, troca de mensagens etc.

Figura 9.5: Blocos avançados (Advanced Blocks)

## 9.6 Meus blocos (My Blocks)

O *MyBlocks* trata-se de categoria especial, onde várias combinações de blocos criados podem ser armazenadas. É possível transformar uma sequência de blocos utilizada no programa em um só bloco, que ficará no *My Blocks*, o que é extremamente útil quando é necessário aproveitar as mesmas sequências de blocos, pois assim não será preciso reescrevê-los.

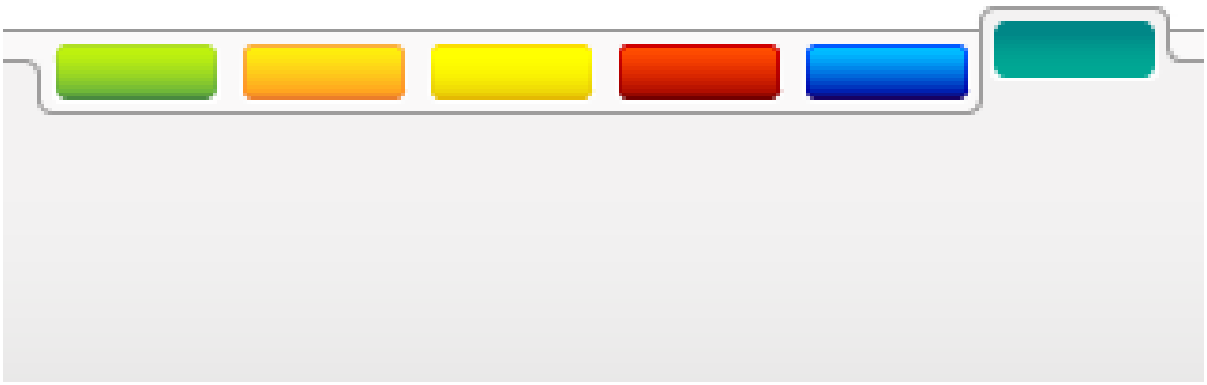


Figura 9.6: Meus blocos (My Blocks)

# Capítulo 10
# Utilização e configuração dos blocos

Neste capítulo, os blocos serão abordados com mais profundidade. Vamos aprender como utilizá-los, e o mais importante: como configurá-los. Para isso, serão trabalhados os principais blocos do programa.

Um bloco possui uma função específica, mas isso não significa que ele só faz uma coisa. Para entender melhor as especificidades de cada um e seus modos de funcionamento, vamos fazer uma análise da constituição de um bloco.

Os **modos** são a forma como o bloco vai trabalhar. Cada modo possui **parâmetros** ou não, sendo que a maioria possui **entradas** e **saídas**. Quando possui parâmetros, deve-se configurá-los de acordo com as necessidades do programa, conforme o tipo de dados que o bloco suporta. Algumas entradas só funcionam com dados textuais, ou com dados lógicos, algumas com mais de um tipo de dado. Isso é uma característica particular de cada bloco. Mas não se preocupe, tudo isso será visto com calma. Nas entradas pode-se conectar um fio de dados, ou apenas estabelecer um valor. Nas saídas estão os resultados das ações dos blocos, sendo possível enviá-las por meio do seu respectivo fio de dados.

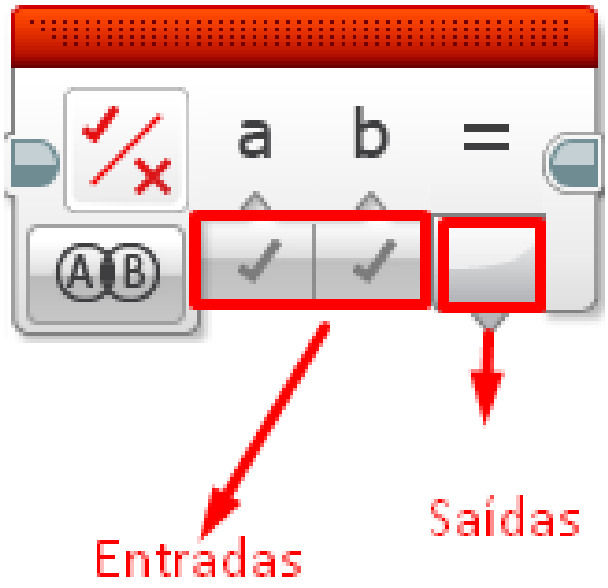



Figura 10.1: Entradas e saídas

Este capítulo aborda a parte teórica. Após a compreensão dos principais blocos será abordada a programação em si, com bastantes exemplos de programas, acompanhados das respectivas explicações e comentários.

Até aqui, já vimos um pouco sobre as paletas de programação e as funções de cada uma, de uma maneira bem geral. Agora abordaremos as configurações que são comuns a quase todos os blocos, mostrando algumas possibilidades e principalmente seus modos e parâmetros. Focaremos nos blocos principais, que são os mais utilizados.

## 10.1 Comunicação entre os blocos

Ao escrever um programa, é necessário realizar a comunicação dos dados entre os blocos para que eles possam partilhar informações, e realizar suas funções. Muitos blocos necessitam de dados de

outros blocos para funcionarem. Por exemplo, usam-se sensores para obter informações do ambiente. Mas o que fazer com essas informações? Devemos usá-las para algum objetivo, como fornecer essa informação a um bloco comparador. Os **Data Wires** (ou fio de dados) são usados para essa função de comunicação entre os dados.

## Data Wires (fios de dados)

É com os fios de dados que as informações são enviadas de um bloco para outro, ou outros. Há os seguintes tipos de fios:

**Fios de dados numéricos:** transmitem números, valores numéricos. São identificados pela cor amarela.

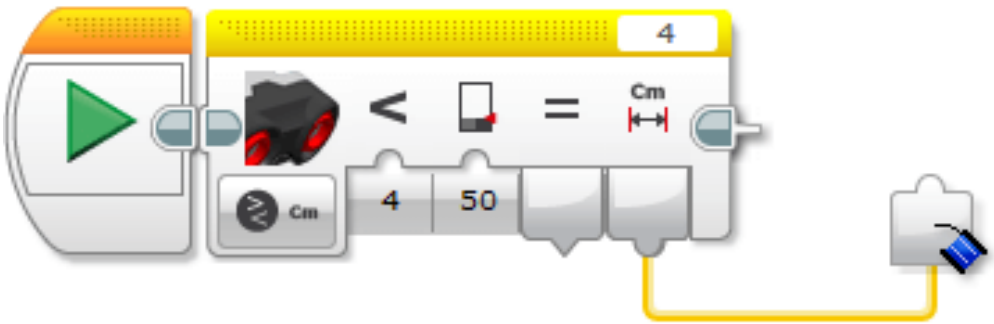



Figura 10.2: Fio de dado do tipo numérico

**Fios de dados lógicos:** enviam dados lógicos, como: verdadeiro/falso. São identificados pela cor verde.

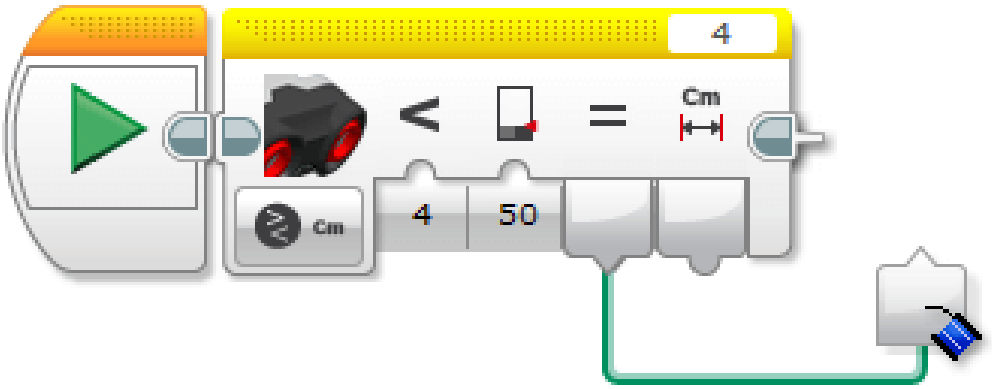



Figura 10.3: Fio de dado do tipo lógico

**Fios de dados textuais:** enviam dados textuais, como palavras, por exemplo. São identificados pela cor laranja.

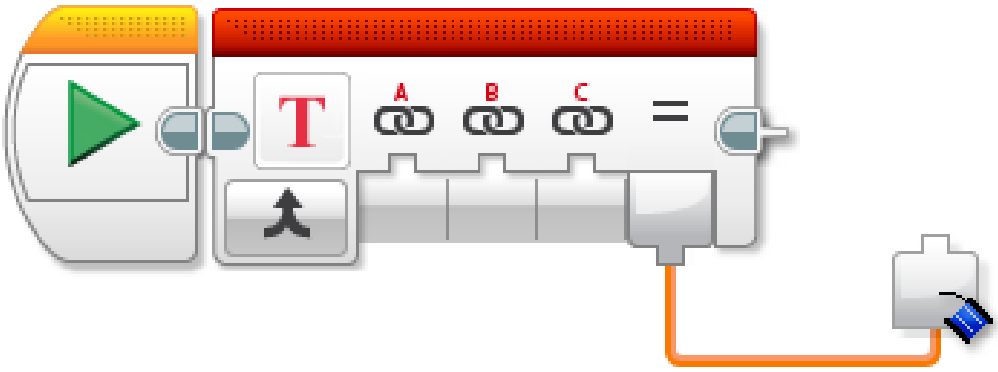



Figura 10.4: Fio de dado do tipo textual

No software LEGO® MINDSTORMS® NXT 2.0 é exibido um fio cinza pontilhado, quando uma ligação incorreta é feita. Quando se comunicam dados de um tipo para uma entrada que foi feita para outro tipo de dado, o software alerta sobre a incompatibilidade, exibindo o fio na cor cinza. Um erro desse impossibilita o download do código.

O software LEGO® EV3 simplesmente não permite esse erro. No software não é possível comunicar um dado para uma entrada que não foi feita para ele. Portanto, quando não for possível realizar a

comunicação, é porque provavelmente há um equívoco relacionado à compatibilidade de dados, devido ao engano cometido sobre o destino do fio, que foi enviado para entrada incorreta.

A conexão a seguir, por exemplo, não seria possível, pois a saída do bloco fornece um dado numérico, mas a entrada do `Switch` é uma entrada de dado do tipo lógica.

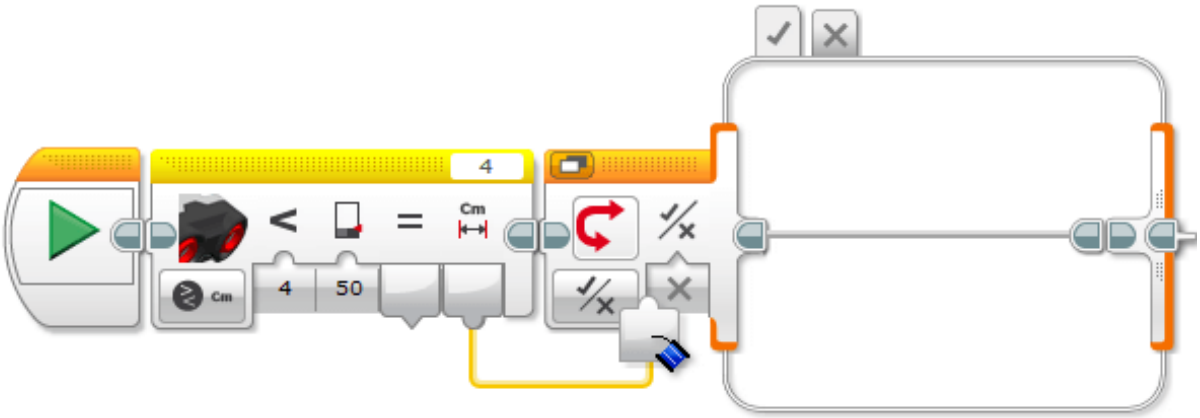



Figura 10.5: Exemplo de conexão inválida

## Conclusão

Conhecer os blocos é o primeiro passo para que você consiga alcançar suas metas e colocar em prática a suas ideias com o seu kit. Com um conhecimento sobre os blocos disponíveis, e como eles funcionam, você já pode começar a esboçar seus próprios programas. Compreender os modos, e como se dá a conexão entre blocos, tendo em mente os tipos existentes de dados, é primordial na programação. Porém, isso representa bem mais que simples conceitos teóricos e regras de sintaxe.

Com esses conhecimetos teóricos você pode elaborar e desenvolver as soluções mais adequadas para a situação. Além disso, você entenderá o seu programa, portanto, não é necessário preocupar-se em gravar esses fundamentos. Todos esses conhecimentos, aqui apresentados, serão abordados e trabalhados na prática.

# Capítulo 11
# Criando e executando um programa no Brick EV3

Estamos quase encerrando mais uma parte do livro. Na próxima parte você finalmente vai criar seus primeiros programas. Por isso, prepare-se e preste bastante atenção neste capítulo. Aqui você vai aprender como criar um novo programa no seu software, como salvá-lo e, por fim, como fazer o download dele para o seu Brick.

Como a versão Home Edition e a versão Education são muito similares (você terá a oportunidade de comprovar essa similaridade pelas imagens), as explicações serão realizadas simultaneamente.

Este capítulo fornecerá um simples e breve exemplo de como criar e executar um programa. Isso pode ser feito de duas formas.

A primeira delas é obtida clicando no sinal  à direita do *Lobby*.

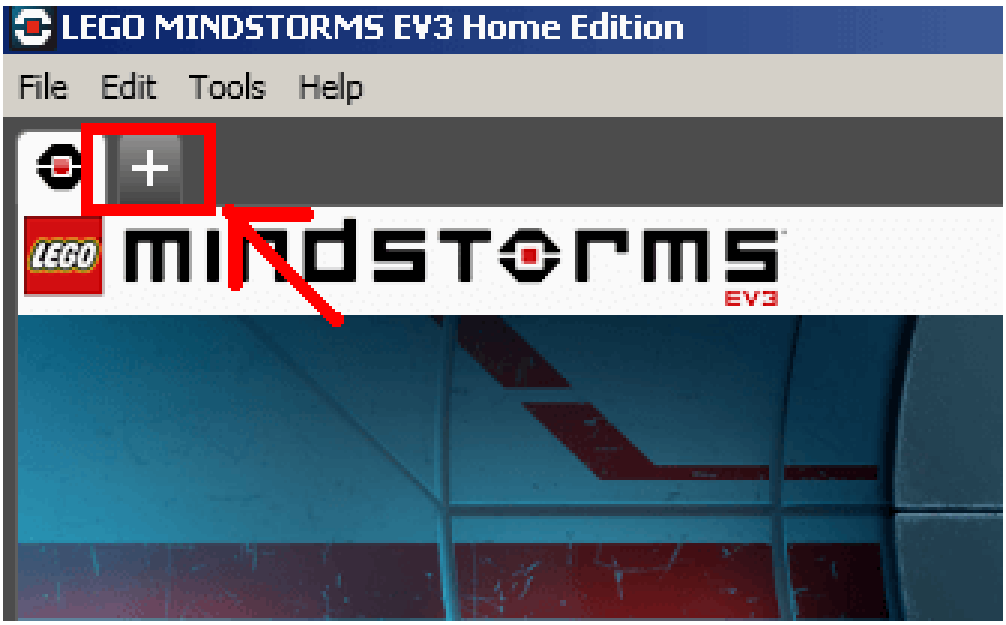



Figura 11.1: LEGO® MINDSTORMS® EV3 Home Edition - Criando um novo projeto / Home Edition

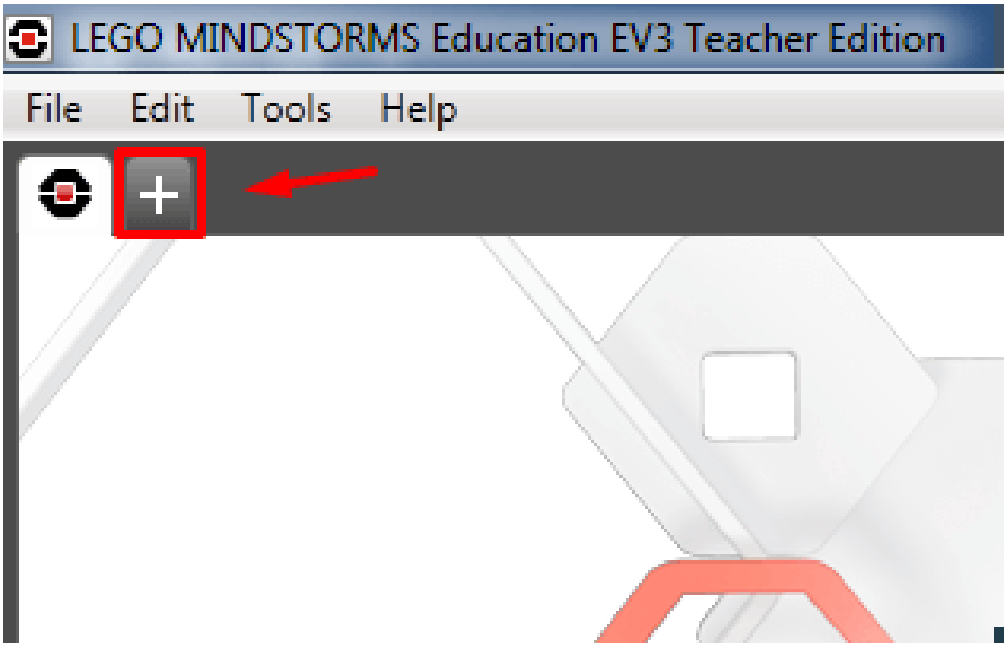



Figura 11.2: LEGO® MINDSTORMS® EV3 Education - Criando um novo projeto / Education

Outra forma de criar um programa consiste em pressionar o botão `File` situado no canto superior esquerdo e, a seguir, clicar na opção `New Project` .

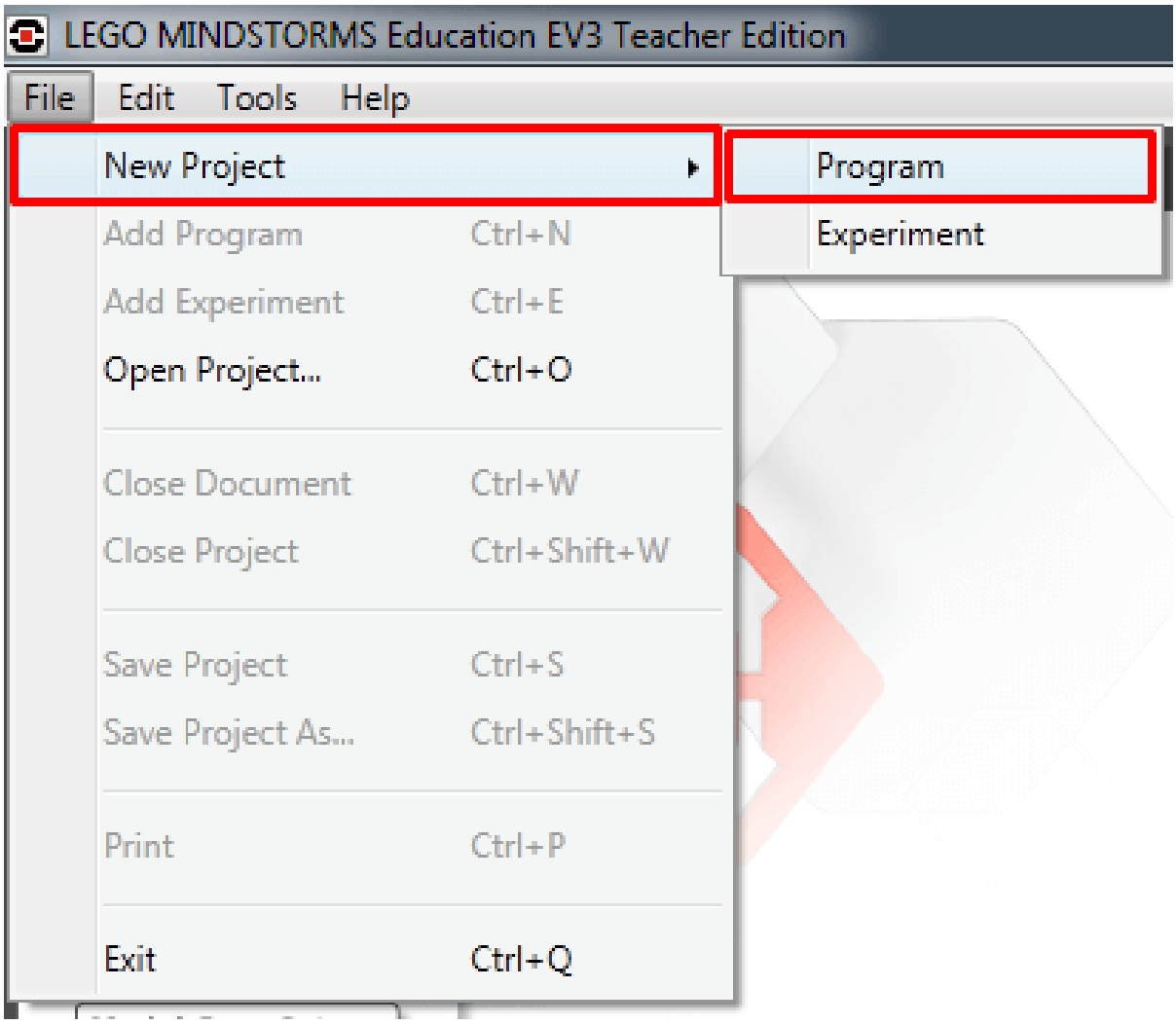



Figura 11.4: LEGO® MINDSTORMS® EV3 Education - Criando um novo projeto / Education

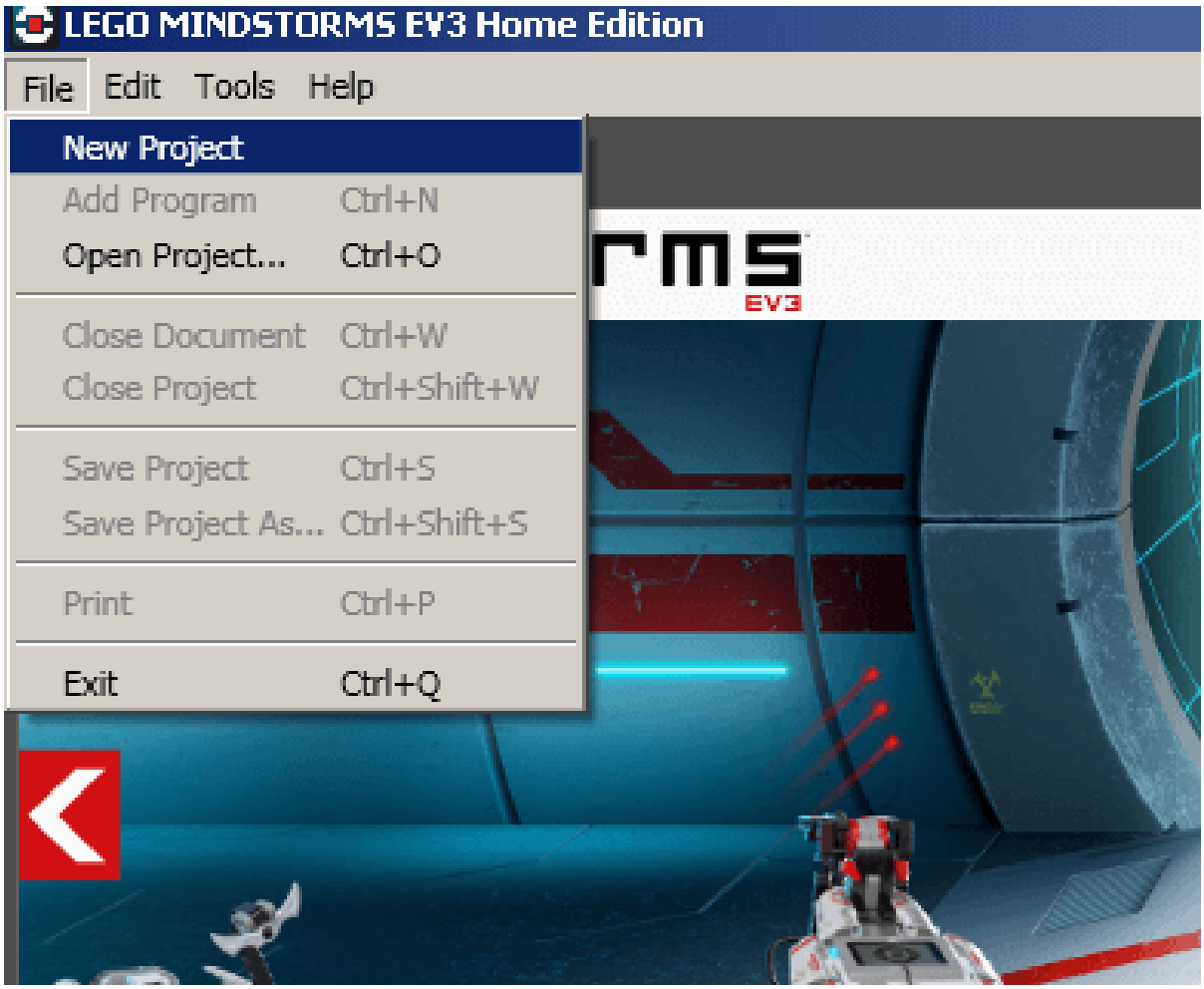



Figura 11.3: LEGO® MINDSTORMS® EV3 Home Edition - Criando um novo projeto / Home Edition

Após a criação de um novo projeto, aparecerá a seguinte tela, na versão Home Edition (na versão Education, a tela é praticamente igual):

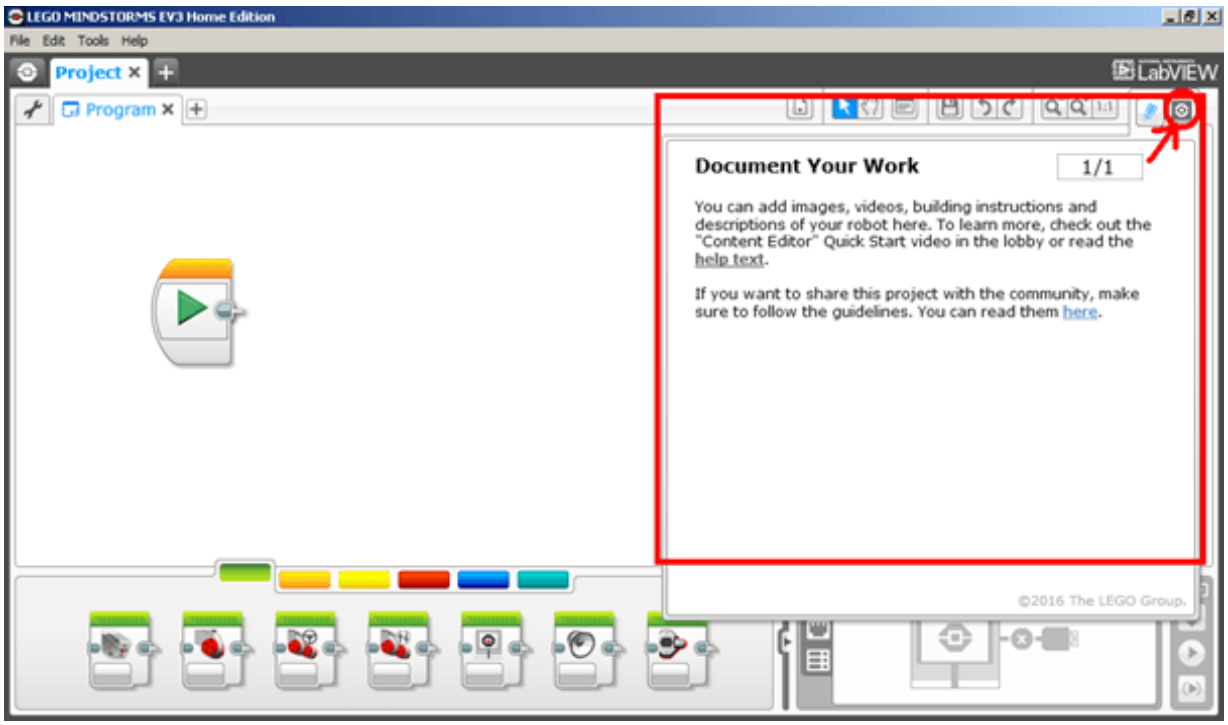

LEGO MINDSTORMS EV3 Home Edition
File Edit Tools Help
Project
LabVIEW
Program
Document Your Work
1/1
You can add images, videos, building instructions and descriptions of your robot here. To learn more, check out the "Content Editor" Quick Start video in the lobby or read the help text.
If you want to share this project with the community, make sure to follow the guidelines. You can read them here.
©2016 The LEGO Group.

Figura 11.5: LEGO® MINDSTORMS® EV3 Home Edition - Tela Inicial do Novo Projeto e Editor de Conteúdos

Quando um novo projeto é aberto, o Editor de Conteúdo inicia-se automaticamente; nele podem ser feitas algumas anotações úteis para explicar o funcionamento do programa, ou algo do tipo. Para fechá-lo basta clicar no botão `Close the Editor` .

Para exemplificar como criar e executar um programa no Brick EV3, vamos utilizar um programa bem simples, que requer apenas o Brick. Imagine que você quer exibir uma palavra na tela do seu Brick, a seguinte expressão: "Oi!". Talvez você já até imagine quais blocos utilizar.

Para exibir uma mensagem na tela do Brick utiliza-se o bloco `Display` . Para exibir a mensagem eternamente, deve-se colocar esse bloco em um `Loop` . Observe na imagem, a seguir, como ficou o programa.

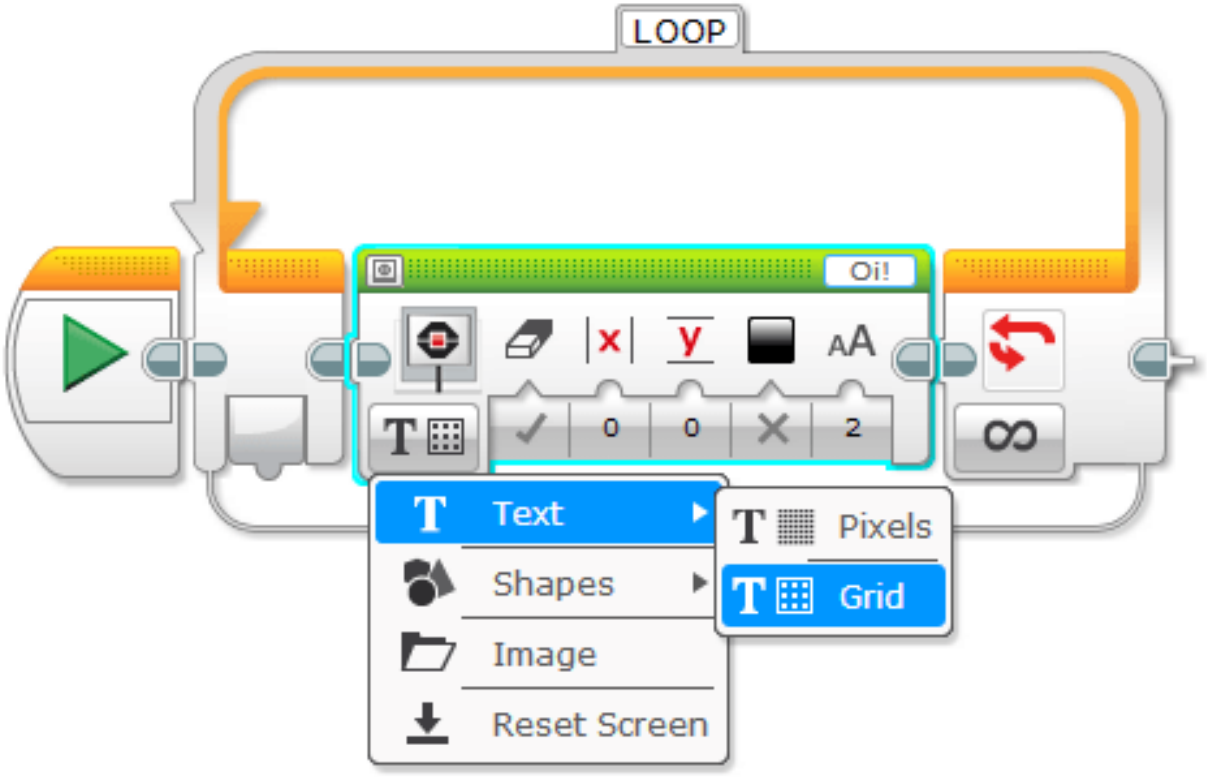



Figura 11.6: Exibindo um "Oi!" na tela

Para não correr risco de perder o programa, por causa de um evento infortuno e inesperado, sempre o salve. O procedimento é o mesmo para a versão Home Edition e para a versão Education.

No canto superior esquerdo do software, clique em `File` , em seguida, escolha entre as opções: `Save Project` ou `Save Project As` . Ambas as opções salvam seu projeto. A opção `Save Project As` é muito útil quando se realiza uma modificação em um projeto anterior e pretende-se salvar o novo sem perder o antigo. Nesse caso, você salva o projeto com um novo nome, sem sobrescrever o anterior. Já a opção `Save Project` simplesmente vai salvando as alterações realizadas no projeto, sobrescrevendo-o.

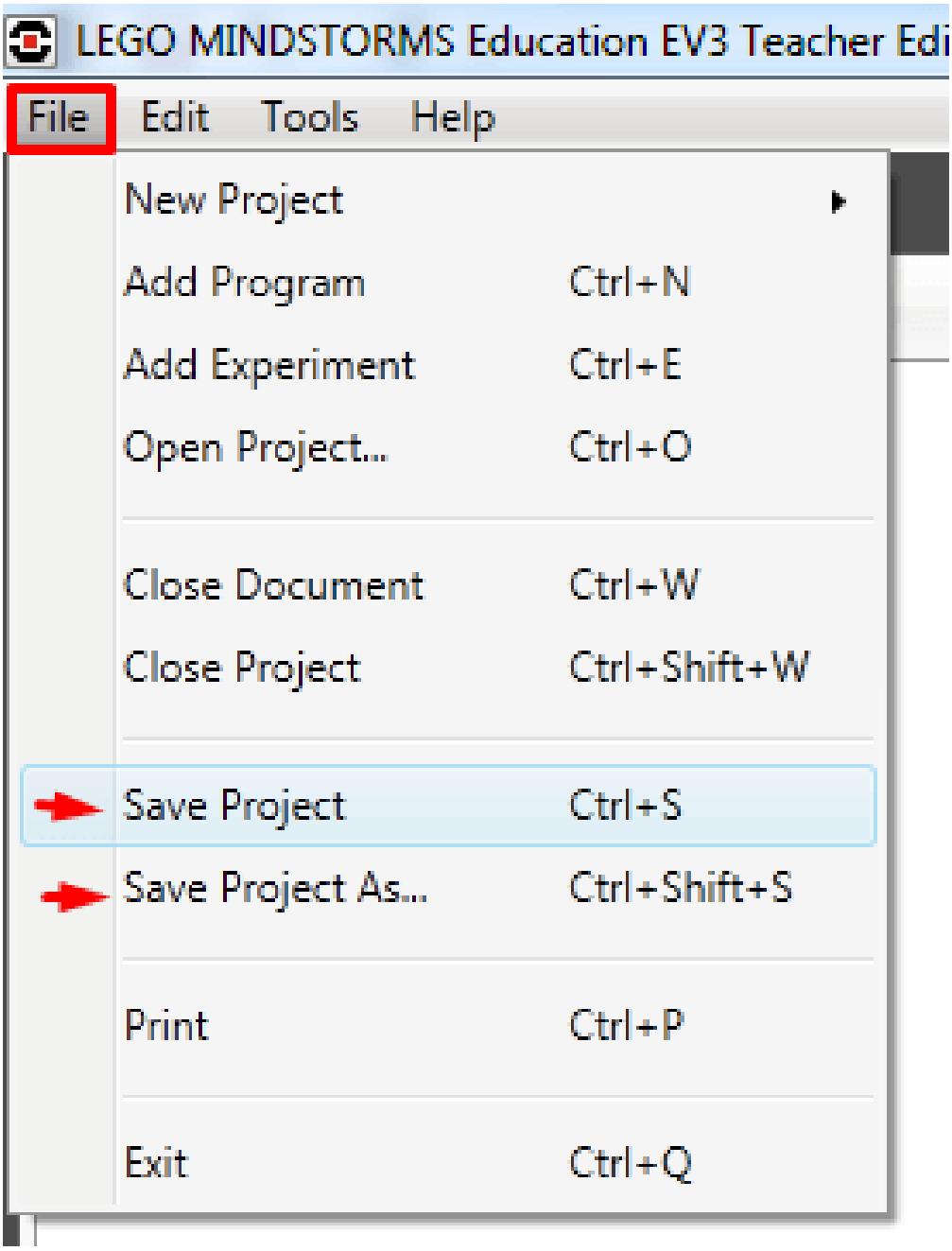



Figura 11.7: Salvando o projeto - Versão Education

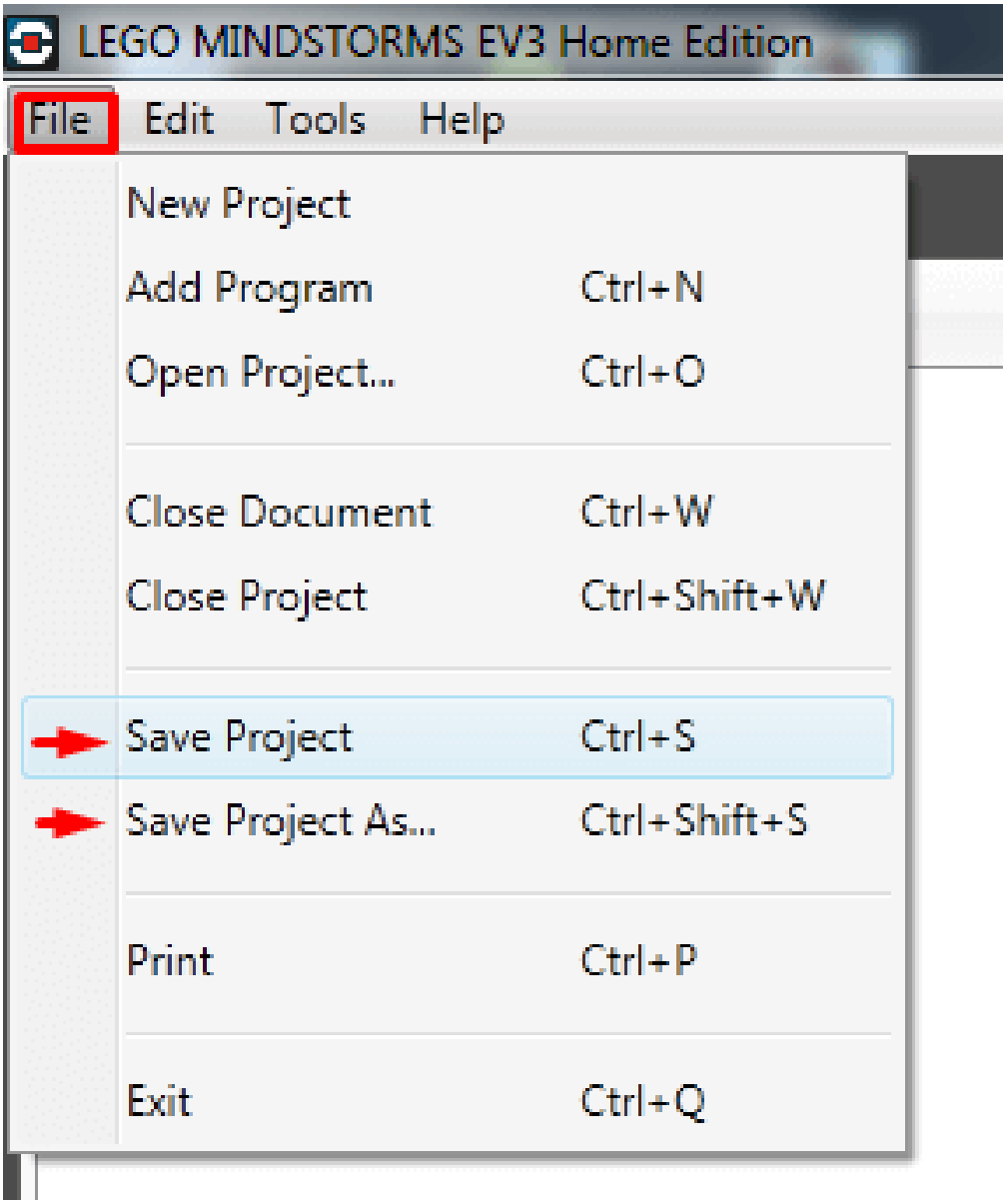



Figura 11.8: Salvando o projeto - Versão Home Edition

Após clicar na opção desejada, uma tela é exibida, nela você escolhe o diretório (local) do computador onde deseja salvar o arquivo. Os arquivos são salvos no formato . ev3 . Caso você não consiga salvar o projeto, pode ser que esteja usando um caractere inválido no nome. Por exemplo, não se deve utilizar o caractere * (asterisco), pois é um caractere reservado. Uma dica muito importante é sempre utilizar nomes curtos e claros para designar o projeto.

Após salvar (ou até mesmo antes), conecte o seu Brick à sua máquina. Para que o programa reconheça o Brick, ele deve estar ligado.

Verifique na *Hardware Page* (no canto inferior direito) se a conexão foi feita. Após isso, faça o download do programa para o Brick.

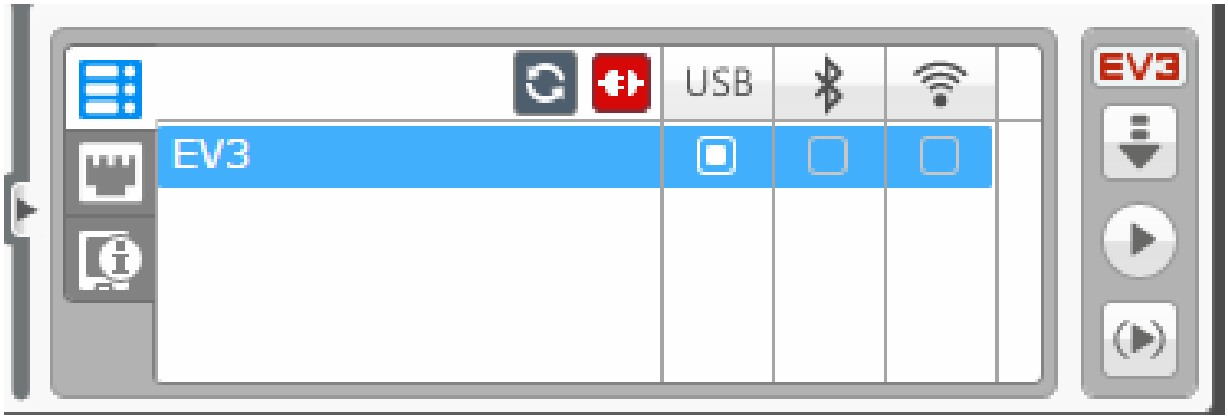



Figura 11.9: Hardware Page

## Conclusão

Agora você já sabe como fazer download dos seus programas para o Brick. Você também aprendeu a diferença entre a forma de salvamento `Save` e `Save As` . Conforme vimos, a forma `Save As` é de grande auxílio, pois assim podemos realizar modificações em um projeto sem sobrescrever o original, e ao longo do livro você perceberá que isso é muito útil. Neste capítulo, você também aprendeu a identificar uma das possíveis falhas que podem ocorrer na hora de fazer o download. E essa falha é evitada de forma bem simples: não usar o caractere `*` (asterisco).

# Capítulo 12
# Conclusão

Se você já chegou até esta página, prepare-se, pois nos capítulos adiante você vai colocar a mão na massa e programar. Os capítulos estudados até aqui forneceram uma prévia do que será visto durante o decorrer do livro. Você já aprendeu sobre os blocos e como eles se encontram dispostos no software. Também já sabe sobre a constituição de um bloco, o que cada um apresenta, seus **modos** de funcionamento, **entradas** e **saídas**. Ao abordar a comunicação dos blocos, vimos que é possível tornar o programa dinâmico, e conhecemos um conceito muito importante: o fio de dados, ou `Data Wires` . Você aprendeu que cada cor (do fio) representa um tipo de conexão. Por último, você viu como salvar seu programa e realizar o download dele para o Brick. Após ver tudo isso, você está mais do que preparado para programar.

> **Dica**
>
> Para concluir esta parte, uma última observação será feita. Desde o início deste livro, as diferenças entre as versões Home Edition e Education estão sendo tratadas, principalmente a respeito de cada ambiente de programação. Porém, nada impede que você mude de plataforma e opte por utilizar uma outra versão. Você deve escolher o ambiente que melhor se encaixa de acordo com suas necessidades.

# PRIMEIROS PROGRAMAS

Aqui se inicia de fato a programação: finalmente serão postos em prática os conhecimentos teóricos obtidos. Inicialmente, vamos abordar projetos bem simples, tanto do ponto vista da programação, quanto da montagem. Esses projetos podem ser fundamentais, e fazer toda a diferença na compreensão dos leitores que não tenham experiência prévia em programação.

Serão elaborados os seguintes projetos:

1. Medidor de distâncias
2. Gerador de números aleatórios
3. Identificador de cores
4. Medidor de luz
5. Detector de presença

Todos os programas que serão feitos neste capítulo estão disponíveis na pasta "Primeiros Programas" (do repositório do livro), que pode ser acessada no seguinte link: https://github.com/vitor000x/ev3_Codes/tree/master/Primeiros-Programas. Logo, caso ao final de um programa você possua alguma dúvida quanto à configuração de um bloco, ou algum outro motivo, fique à vontade para conferir os programas.

> **Observação**
>
> Alguns `Switchs` do programa foram configurados no modo de exibição `Tabbed View` , para que a visualização do programa fosse facilitada.

## Capítulo 13
## Medidor de distâncias

Como não existe o sensor ultrassônico na Home Edition, este projeto foi desenvolvido tanto com o sensor ultrassônico quanto com o sensor infravermelho, assim você poderá implementá-lo com qualquer versão. Por isso, este capítulo é divido em duas partes.

É ideal que o leitor usuário da versão Home Edition acompanhe o primeiro programa feito para a versão Education, para que ele compreenda a lógica. A construção deste projeto com o sensor Infravermelho envolve apenas a modificação de um bloco. Portanto, mesmo se você possui a versão Home Edition, não deixe de conferir a construção deste programa.

Neste primeiro programa, você terá a oportunidade de aplicar a teoria que vimos até agora na prática. Ele tem como objetivo medir a distância do sensor ultrassônico até um objeto situado à sua frente.

## 13.1 Medidor de distâncias com sensor ultrassônico

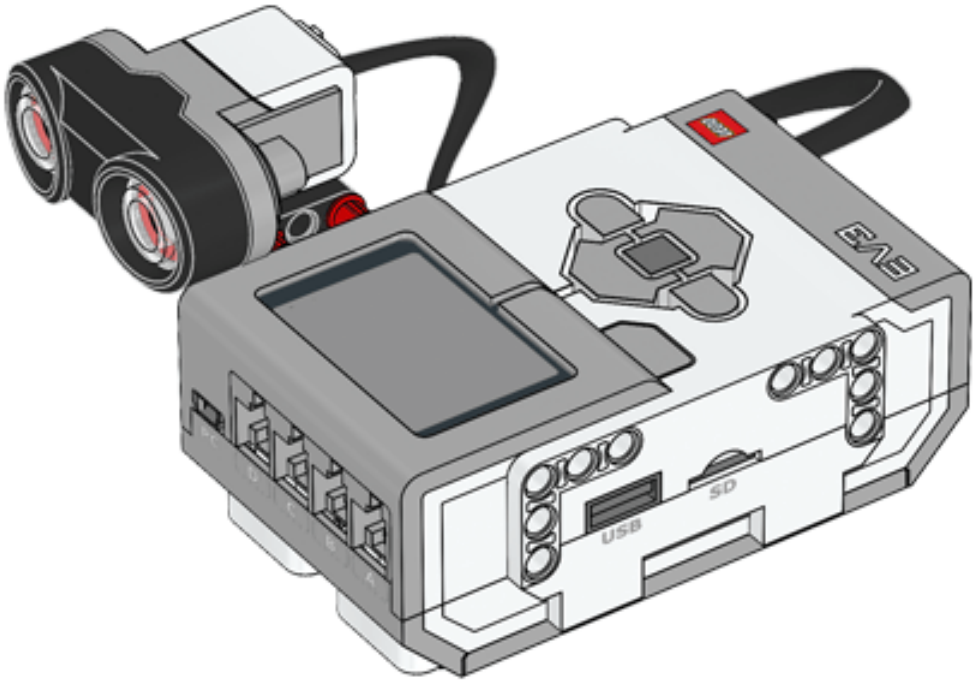



Figura 13.1: Medidor de distâncias com sensor ultrassônico

## Instruções de montagem

Este projeto requer apenas o Brick e um sensor ultrassônico conectado a ele. Porém, caso surja alguma dúvida quanto à montagem, e o usuário esteja trabalhando com a edição Education do EV3, siga os seguintes passos: No *Lobby*, localize a opção `Building Instructions` , em seguida, clique em `Building Ideas` e procure pelo `Ultrasonic Sensor - Brick` . Clique nele e, por fim, clique em `Open` para abrir o manual. A imagem a seguir ilustra isso.

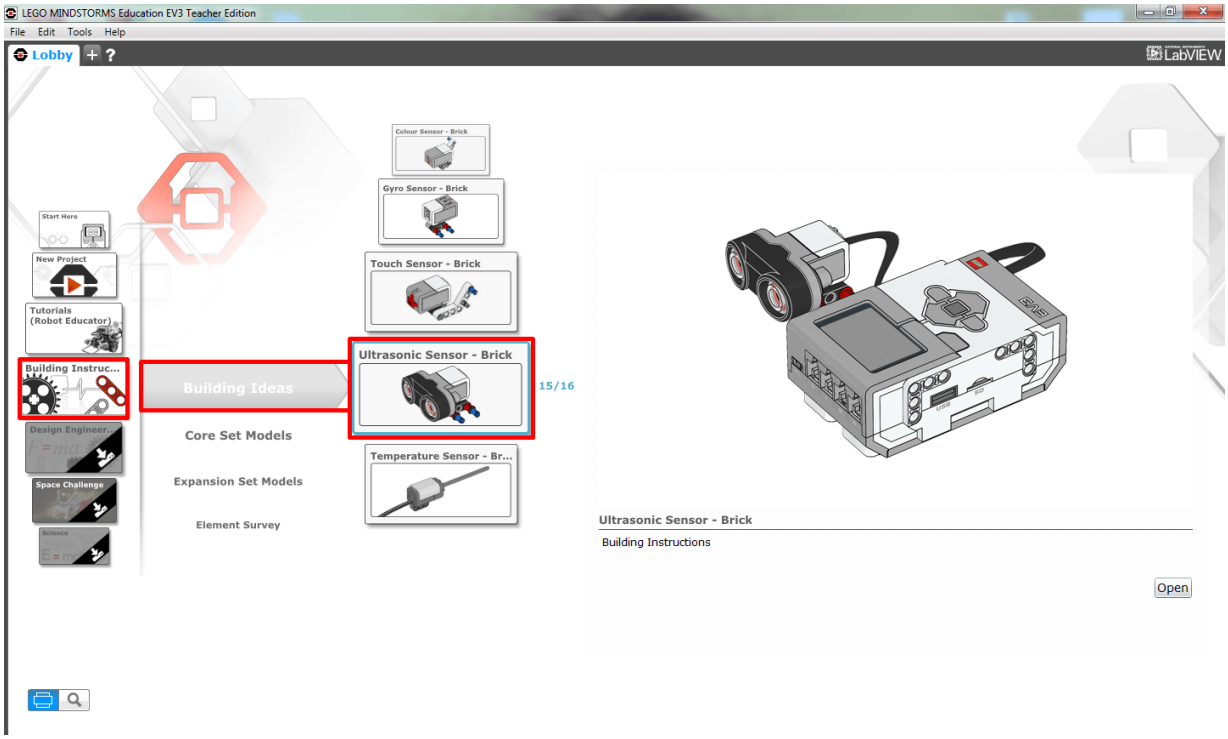



Figura 13.2: Medidor de distâncias com sensor ultrassônico - Instruções (Teacher Edition)

Caso você ainda não esteja utilizando a versão atualizada do software, deve-se acessar o *Lobby* e clicar em `Robot Educator` , `Building Instructions` , `Ultrasonic Sensor – Brick` e, finalmente, em `Open` , para ter acesso às instruções de montagem.

## Objetivo do programa

Este programa exibirá a mensagem “Go!” no display quando iniciado. Caso o botão do meio do Brick seja pressionado, o sensor ultrassônico a ele conectado realizará a medição, e exibirá o valor no display por dois segundos. Após isso, volta-se ao começo do programa, cujo código é envolvido por um `Loop` , configurado no modo `Unlimited` (loop infinito).

## O código

Como demonstrado anteriormente, todo programa inicia-se com um bloco `Start` . Após o bloco `Start` , tem-se o bloco `Loop` . Este `Loop` tem a

função de repetir as ações contidas nele, sendo configurado no modo `Unlimited` (ilimitado) para repetir a ação infinitamente. Dentro do `Loop` , o primeiro bloco é o `Display` , responsável por controlar o visor do Brick. Com ele, é possível exibir imagens, textos, números e formas. Ainda é possível configurar em qual lugar específico da tela eles aparecerão; isto é feito por meio de coordenadas, que são configuradas com `X` e `Y` . O `Display` apresenta 177 pixels de largura por 127 de altura. `X` corresponde a largura e `Y` a altura.

No programa nós utilizamos o `Display` configurado no modo `Text` e, em seguida, escolhemos a opção `Grid` . Mas você também pode utilizar a opção `Pixels` . Talvez você tenha reparado que, na opção `Grid` , as coordenadas `X` e `Y` apresentam, respectivamente, os seguintes valores: 21 e 12. O modo `Grid` faz com que o texto fique alinhado à grade do `Display` . Repare que, se você passar o mouse sobre o parâmetro `X` , vai aparecer a palavra `Column` , e sobre o `Y` , aparece `Row` . Isso acontece pois, quando utilizamos este modo de exibição, as referências não são mais os `Pixels` do `Display` , passamos a ter como referência as colunas ( `Column` , letra `X` ) e linhas ( `Row` , letra `Y` ), que formam a grade, na qual o texto é alinhado.

## 13.2 O bloco Loop

Até aqui vimos um pouquinho do bloco `Loop` . Mas, talvez, você tenha se perguntado: o bloco `Loop` sempre repete uma ação infinitas vezes? Aqui, vimos apenas como utilizá-lo no modo infinito ( `Unlimited` ), que é um dos modos mais utilizados. Mas ele permite outras configurações. Dentre os principais modos, podem-se destacar:

- `Count` - Com ele, você pode determinar o número de vezes que o `Loop` será executado.

- `Time Indicator` - Com ele, pode-se determinar um tempo de duração do `Loop` em segundos.
- `Logic` - O programa é executado mediante uma condição, “até que”. Pode se escolher executar até que a condição seja verdadeira ( `True` ), ou até que a condição seja falsa ( `False` ).

Esses são os principais modos, mas existem outros, de acordo com os sensores, cujos estados podem ser utilizados como condição de controle do `Loop` . Ao decorrer do livro, trabalharemos os principais modos de operação do bloco `Loop` .

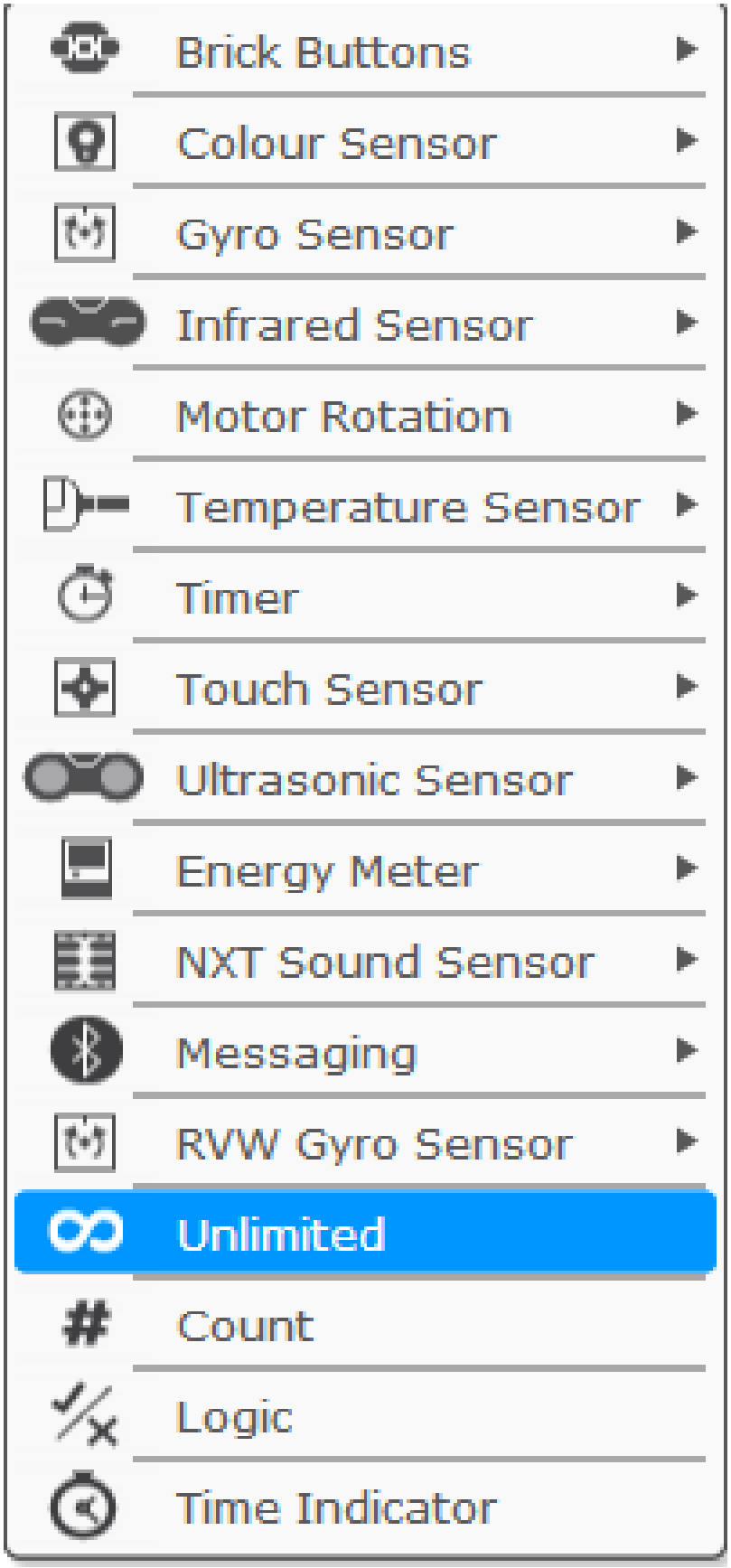

Brick Buttons
Colour Sensor
Gyro Sensor
Infrared Sensor
Motor Rotation
Temperature Sensor
Timer
Touch Sensor
Ultrasonic Sensor
Energy Meter
NXT Sound Sensor
Messaging
RVW Gyro Sensor
Unlimited
Count
Logic
Time Indicator

Figura 13.3: Modos de operação do bloco Loop

## 13.3 Medidor de distâncias

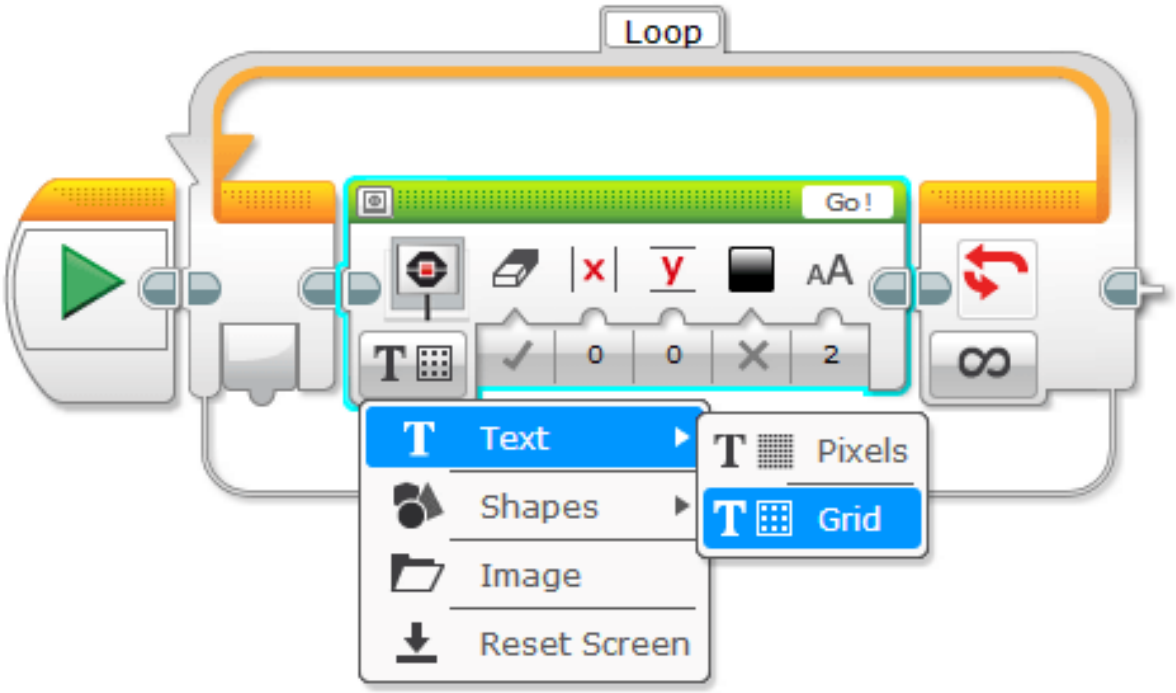



Figura 13.4: Medidor de distâncias - Primeiros passos

Agora inicia-se a parte crucial e mais importante do programa, em que a medição é feita e exibida na tela. Para isso, será utilizado o `Switch` , que deve ser configurado no modo `Brick Buttons` e deve-se escolher a opção `Compare` . Com isso, o `Switch` compara se o botão foi pressionado; caso tenha sido, isso corresponde ao valor lógico verdadeiro, e a linha de cima é executada; caso contrário, nada acontece.

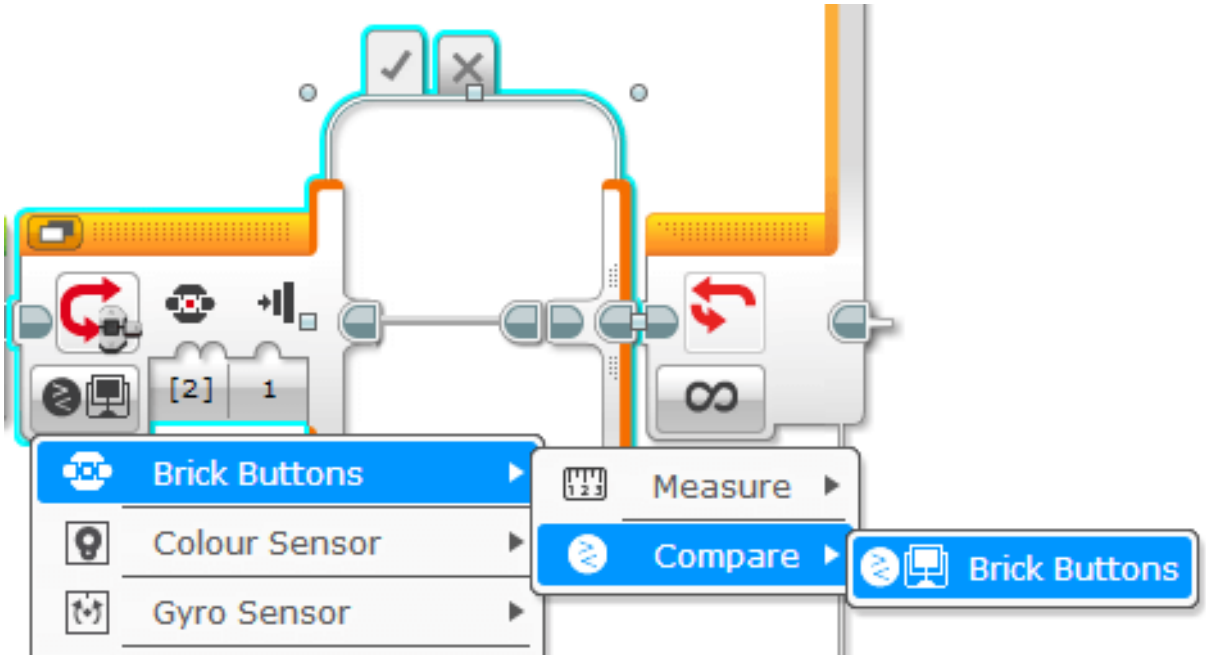



Figura 13.5: Medidor de distâncias - Adicionando o Switch

Dentro do `Switch` , deve ficar o bloco `Ultrasonic Sensor` . Para que a medição seja realizada, ele é configurado da seguinte forma: escolhe-se a opção `Measure` e, em seguida, `Distance in Centimeters` (poderia ser utilizada em polegadas também, escolhendo a opção `Distance in Inches` ).

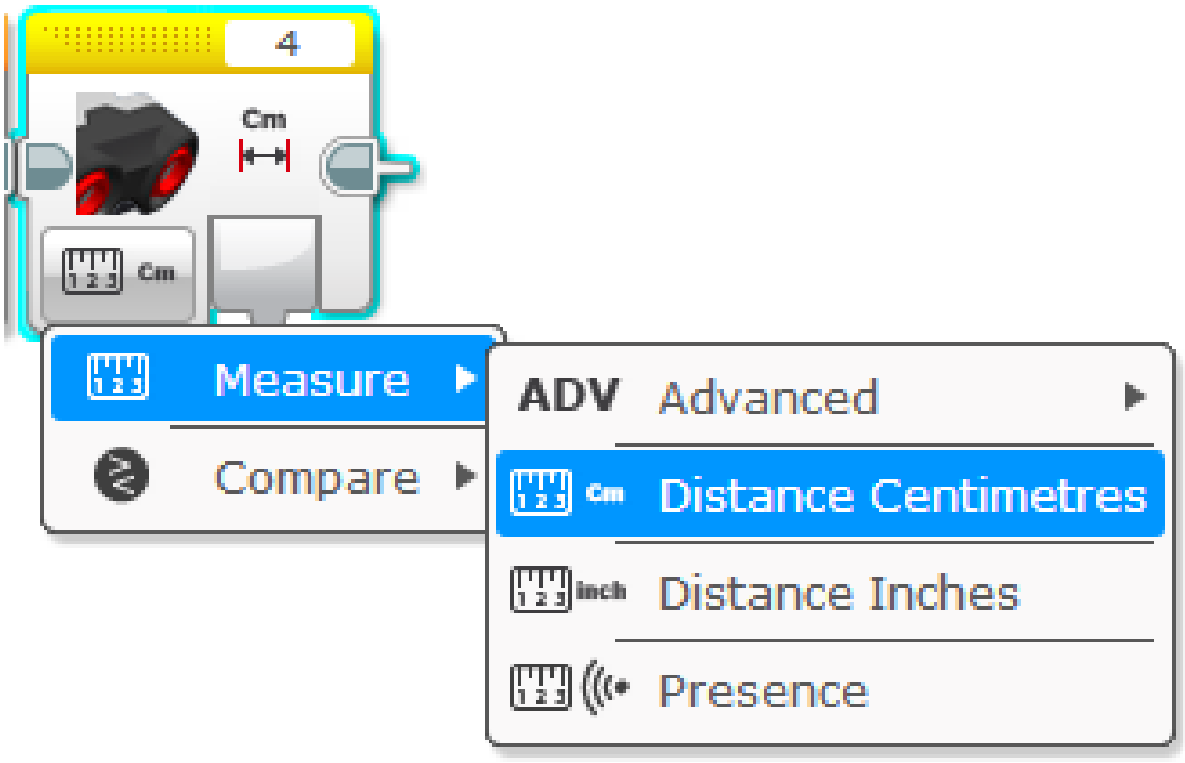



Figura 13.6: Medidor de distâncias - Configurando o sensor ultrassônico

O valor obtido é enviado por meio do fio amarelo para o bloco `Display` . Vale lembrar que é preciso escolher a opção `Wired` no bloco `Display` , para que o conteúdo da tela seja enviado pelo `Data Wires` (ou fio de dados). Por último, temos o bloco `Wait` , que foi configurado no modo `Time Indicator` e ajustado para 2 segundos. Esse bloco de espera faz com que o conteúdo da tela seja exibido por 2 segundos. Se tal bloco não estivesse presente, o conteúdo da tela seria exibido muito rápido, de forma praticamente imperceptível.

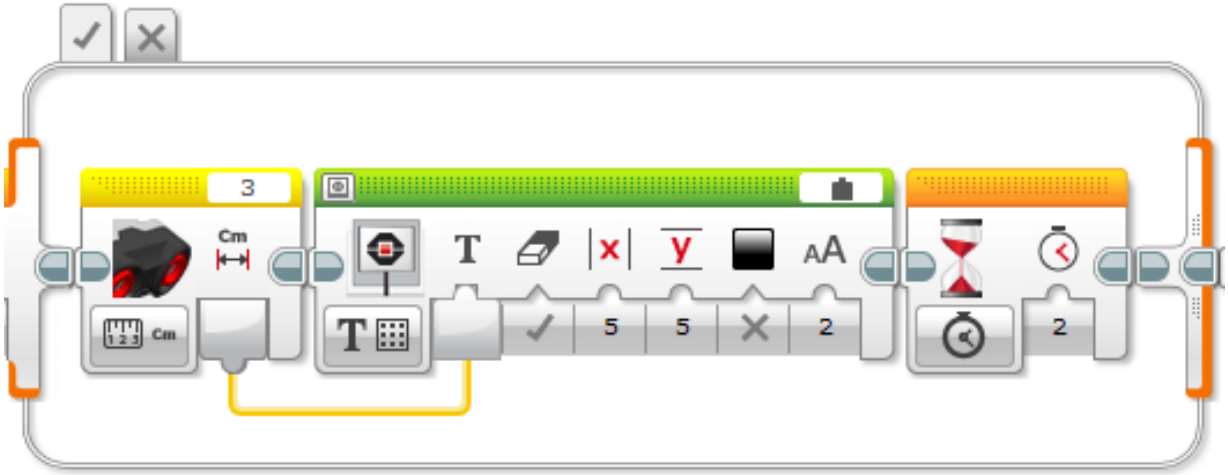



Figura 13.7: Medidor de distâncias - Parte final do programa: medindo a distância e exibindo na tela

## O programa ficou da seguinte forma:

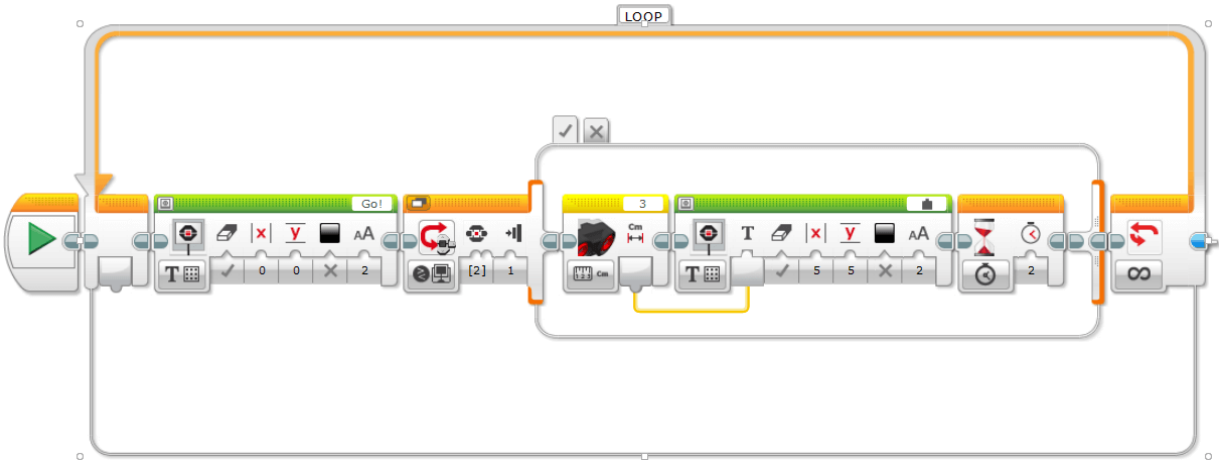



Figura 13.8: Medidor de distâncias - Código completo

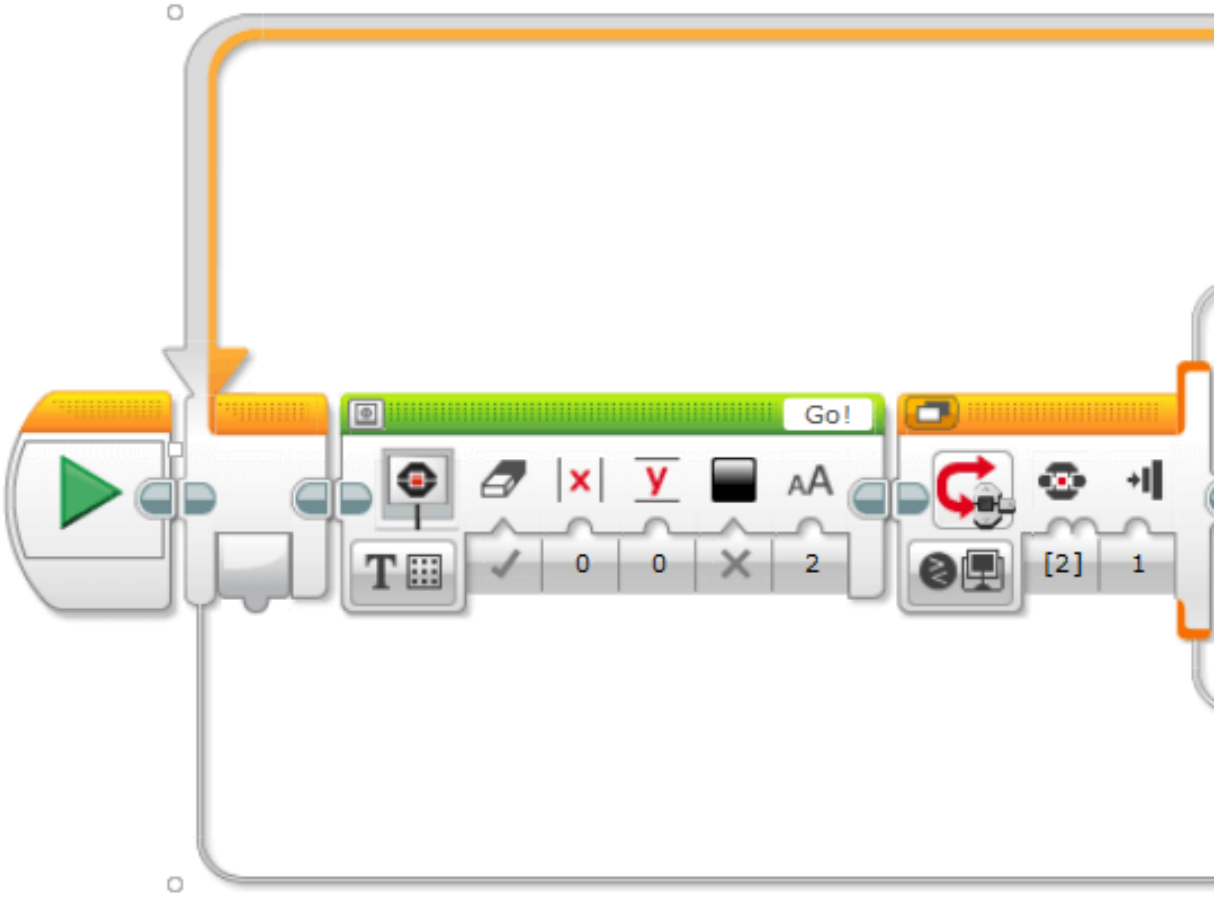



Figura 13.9: Medidor de distâncias - Detalhe 1

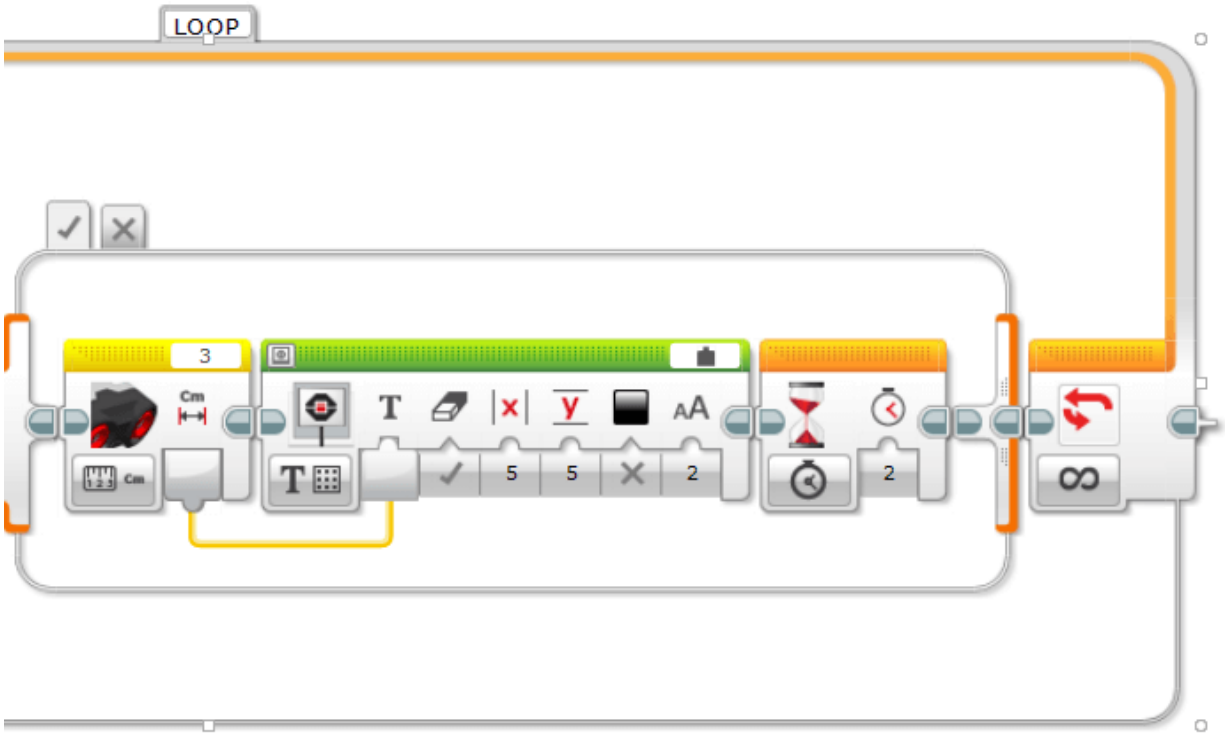



Figura 13.10: Medidor de distâncias - Detalhe 2

## 13.4 Medidor de proximidade com sensor infravermelho

Conforme dito anteriormente, nesta parte será explicado como fazer a adaptação do código para os usuários da versão Home Edition, mas a leitura da seção anterior é importante para que se possa compreender o programa. Quanto às instruções de montagem, o leitor pode seguir as instruções anteriores, mas claro, utilizando o sensor infravermelho em vez do ultrassônico. Logo, com este programa você obterá noção de quão próximo o objeto está do sensor infravermelho.

Uma simples alteração é necessária para que o usuário consiga realizar o projeto. O sensor infravermelho pode ser utilizado para

medir a proximidade de um objeto até ele. Dessa forma, é possível medir o quão próximo estão os objetos que se encontram à frente do sensor. Para isso, utiliza-se o modo `Measure - Proximity` . A figura a seguir ilustra como fazer isso.

> **Observação**
>
> Esta é apenas uma adaptação, logo, a utilização do sensor infravermelho como uma espécie de sensor ultrassônico apenas lhe oferece uma ideia de proximidade, que segue uma escala da 0 a 100.

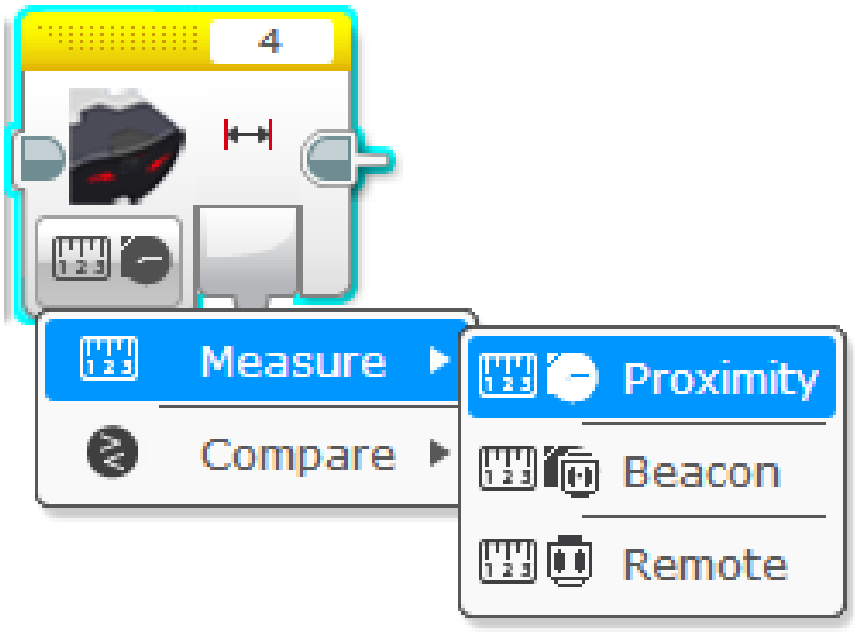



Figura 13.11: Medidor de proximidade - Configurando o sensor infravermelho

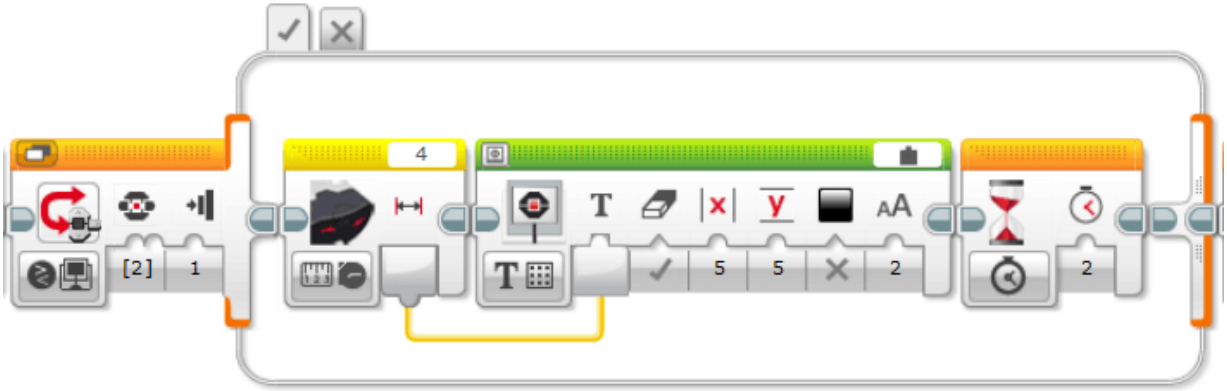



Figura 13.12: Medidor de proximidade - Sensor infravermelho

## 13.5 O bloco Round

Figura 13.13: ícone do bloco Round

Às vezes, durante a execução de um determinado programa, pode ocorrer de se obter números indesejados, como números não inteiros, com casas decimais. Esses números podem exigir uma lógica mais complexa para serem manipulados durante o programa. Com o bloco `Round` é possível arredondar os números com casas decimais.

### Entrada

*Input*: usado para escrever o número que será arredondado. Como é uma entrada, pode-se conectar um fio de dados numérico nela. Por exemplo, é possível enviar o valor da medição do sensor ultrassônico à entrada.

**Saída**

*Result*: pode-se transmitir o resultado da operação a outro bloco.

Apresenta os seguintes modos de funcionamento: *To Nearest*, *Round Up*, *Round Down* e *Truncate*. Todos estes modos apresentam a entrada `Input` e a saída `Result` . O modo *Truncate* ainda conta com uma entrada a mais.

Vamos ver, agora, de forma mais detalhada, os modos de funcionamento deste bloco:

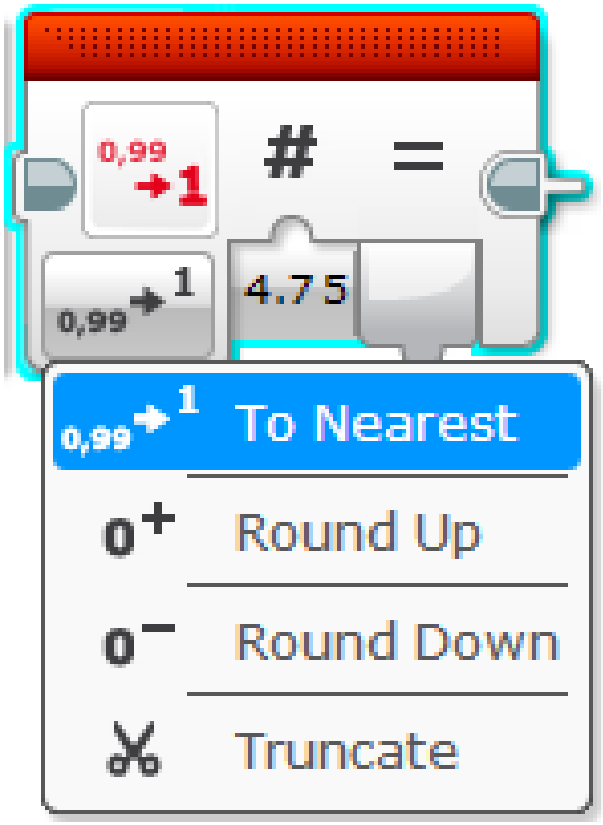



Figura 13.14: Modos de funcionamento do bloco Round

- **To Nearest**

Arredonda seguindo regras de aproximação numérica. Casas decimais maiores ou iguais a 0.5 são arredondas para cima. Casas decimais menores que 0.5 são arredondadas para baixo. Exemplificando:

| **Número (*Input*)** | **Resultado (*Result*)** |
|---|---|
| 1.0 | 1 |
| 1.1 | 1 |
| 1.3 | 1 |
| 1.5 | 2 |
| 1.7 | 2 |

- **Round Up**

Arredonda o número para cima. Exemplificando:

| **Número (*Input*)** | **Resultado (*Result*)** |
|---|---|
| 1.0 | 1 |
| 1.1 | 2 |
| 1.3 | 2 |
| 1.5 | 2 |
| 1.7 | 2 |

- **Round Down**

Arredonda o número para baixo. Exemplificando:

| **Número (*Input*)** | **Resultado (*Result*)** |
|---|---|
| 1.0 | 1 |

| Número (*Input*) | Resultado (*Result*) |
|---|---|
| 1.1 | 1 |
| 1.3 | 1 |
| 1.5 | 1 |
| 1.7 | 1 |

- **Truncate**

Esse modo permite selecionar o número de casas decimais para o número final. Em `Input` é escrito o número que será ajustado. O `Result` obtido será de acordo com o número de casa decimais selecionado. É possível selecionar até 4 casas. Ele apresenta outra entrada além de `Input` .

`Number of Decimals` : aqui se especifica o número de casas decimais desejadas para o número final.

Figura 13.15: Modo Truncate

Exemplificando:

| Número (*Input*) | *Number of Decimals* | Resultado (*Result*) |
|---|---|---|
| 2.321 | 0 | 2 |
| 2.321 | 1 | 2.3 |

| Número (*Input*) | *Number of Decimals* | Resultado (*Result*) |
|---|---|---|
| 2.321 | 2 | 2.32 |

## 13.6 Medidor de distâncias com arredondamento

Acabamos de ver detalhadamente a função do bloco `Round` e seus modos. Agora vamos colocar em prática esses conhecimentos adquiridos.

Incrementaremos no programa "Medidor de distâncias com sensor ultrassônico", a seguinte funcionalidade: o valor medido pelo sensor ultrassônico será arredondado para cima.

Isso é bem simples de se fazer, o primeiro passo é entender como deve ser feita a operação. Em vez de transmitirmos o valor medido diretamente para o bloco `Display` , vamos enviá-lo, antes, para o bloco `Round` , que devemos adicionar ao programa. Conforme vimos, o modo que realiza o arredondamento para cima é o `Round Up` , logo, devemos configurar o bloco `Round` neste modo.

O valor medido pelo sensor ultrassônico (que se encontra na saída `Distance in Centimetres` ) é enviado para a entrada `Input` do bloco `Round` e, em seguida, é só enviar o resultado da operação (contido na saída `Result` ) para a entrada `Text` do bloco `Display` .

Observe na imagem, a seguir, a modificação feita:

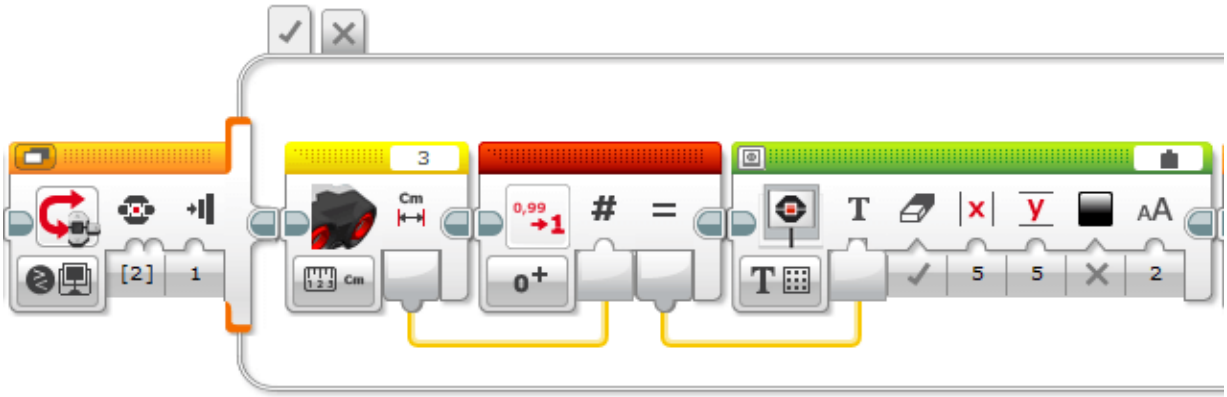



Figura 13.16: Utilizando o bloco Round

Estudamos o conceito teórico e vimos uma aplicação prática do bloco `Round` . Dessa forma, você pôde entender melhor sua utilidade e forma de funcionamento, conhecendo todas as possibilidades que esse bloco permite.

## 13.7 Tente fazer

Com este programa, você acaba de subir mais um degrau na escada de aprendizagem. Por mais que ele possa parecer simples, ele é fundamental para seu aprendizado. Você aprendeu blocos novos, descobriu novos recursos e possibilidades. Porém, para ter um salto ainda maior na sua curva de aprendizado e ganhar cada vez mais experiência é preciso praticar. Até aqui você foi orientado e recebeu instruções. Mas que tal um desafio? Para que você se torne um expert, algumas atividades e sugestões serão recomendadas. Não preocupe, você não terá que criar um programa novo, do zero. Tente fazer algumas pequenas variações e modificações no programa que acabamos de criar. A seguir estão listadas algumas modificações que você pode tentar.

- Você pode optar por exibir uma forma geométrica (círculo, retângulo, triângulo, quadrado) ou uma imagem, em vez de exibir um texto. Para exibir uma forma geométrica, escolha o modo `Shapes` , e para exibir uma imagem, selecione o modo `Image` . Após selecionar o modo, clique em `File Name` . Dessa forma, é possível escolher uma imagem do projeto na pasta `Project Images` (que foi previamente adicionada ao projeto, por meio da ferramenta *Image Editor*, presente na aba *Tools* e que já foi discutida), ou uma imagem enviada através de um Wired, ou uma imagem da pasta `LEGO Images Files` . Nessa pasta você encontra diversas imagens disponibilizadas pelo software, basta clicar na imagem de seu interesse.

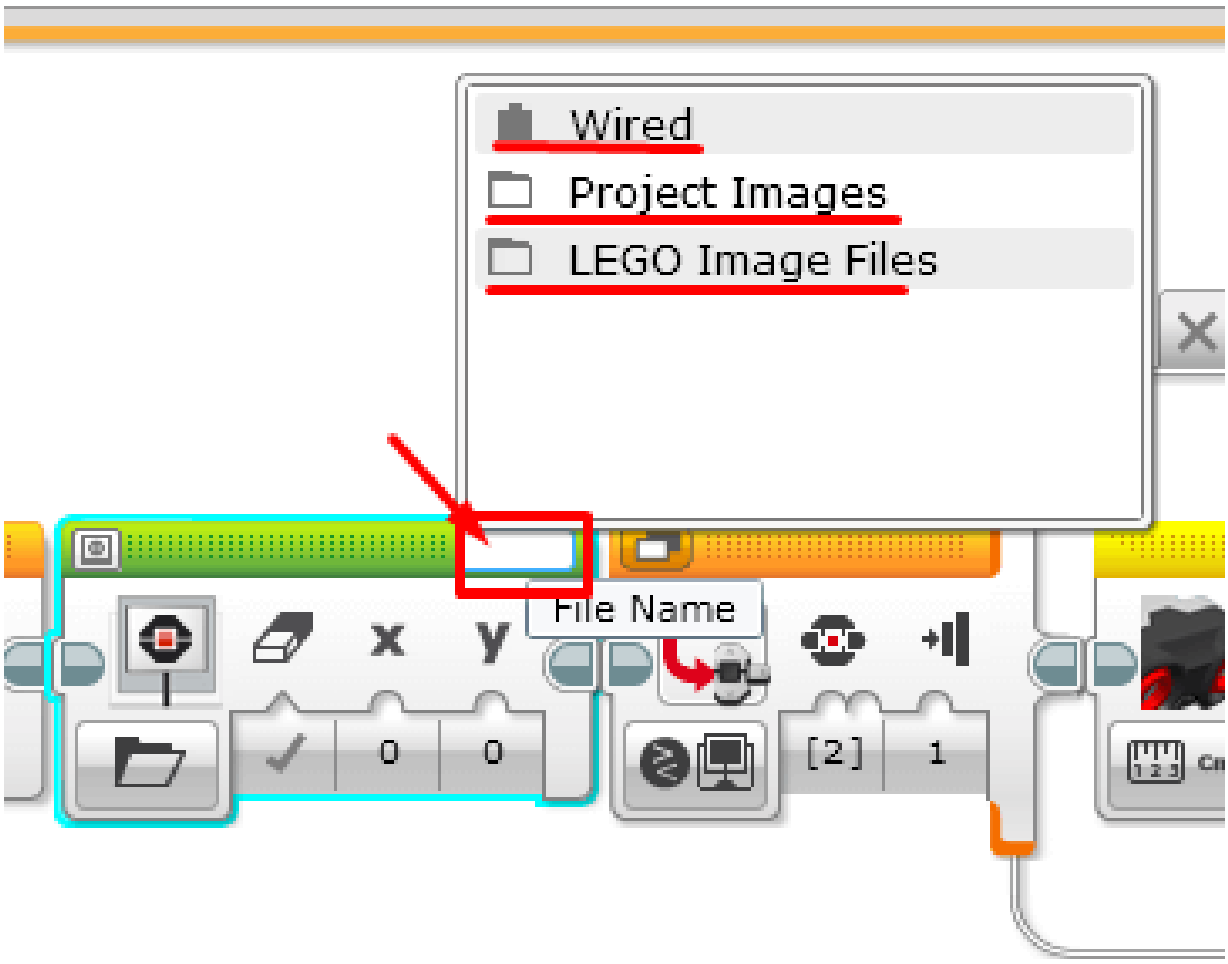



Figura 13.17: Modo Image - Pastas disponíveis

**DICA**

Para visualizar como será a exibição do conteúdo, pode-se utilizar o próprio bloco. Basta clicar no botão situado no canto superior esquerdo, o `Display Preview` . Observe que, se você utilizá-lo para ver algo que está sendo exibido no modo `Grid` , o Display Preview exibirá uma grade, a mesma que foi discutida anteriormente.

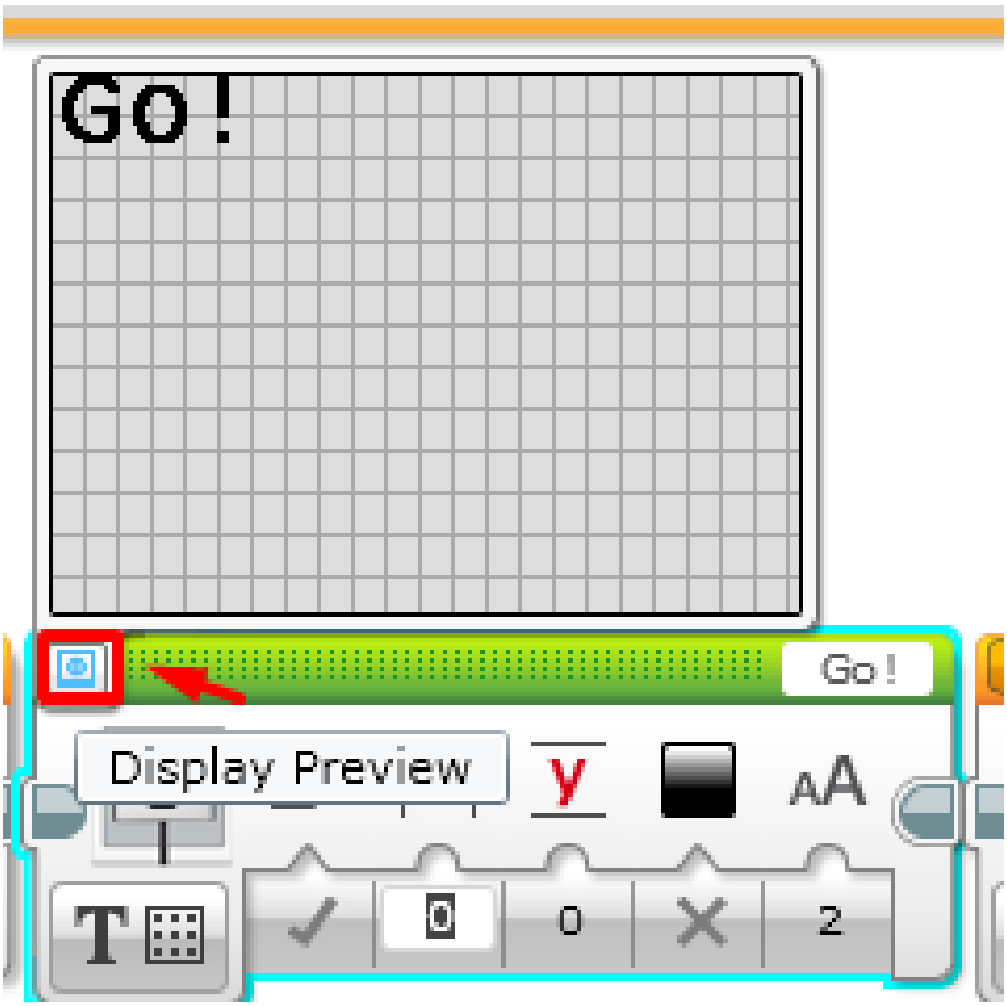



Figura 13.18: Display Preview

# Capítulo 14
# Gerador de números aleatórios

Como o próprio nome já diz, este programa gera números aleatórios. O Brick simula o sorteio de um dado. Aqui vamos sortear um número aleatório entre 1 e 10, mas pode-se facilmente modificar a faixa de números.

## Instruções de montagem

Requer apenas o Brick, não necessitando de nenhum outro componente.

## Objetivo do programa

O programa exibirá no display do Brick a mensagem "Random!". Quando o botão do meio for pressionado, ele gerará um número aleatório entre 1 e 10, que será exibido no display por 3 segundos. Após isso, volta-se ao início e a mensagem “Random!” é exibida. O programa pode executar infinitas vezes, pois está em `Loop - Unlimited` (loop infinito).

## O código

O primeiro bloco que adicionaremos, após o bloco `Start` , é o bloco `Loop` , configurado em `Unlimited` , para executar a ação nele contida infinitas vezes. Dentro do `Loop` vamos adicionar o bloco `Display` , configurado no modo `Text` e na opção `Grid` , com as seguintes coordenadas: x = 5 e y = 5. Contudo, nada impede que você modifique esses valores e exiba os dados na posição do `Display` que mais achar conveniente. O bloco exibe a mensagem “Random!”. Dessa forma, quando o programa for iniciado no Brick, a primeira coisa a ser realizada será a exibição da mensagem "Random!" na tela. Até aqui nada de muito diferente, pois estudamos estes blocos no capítulo anterior.

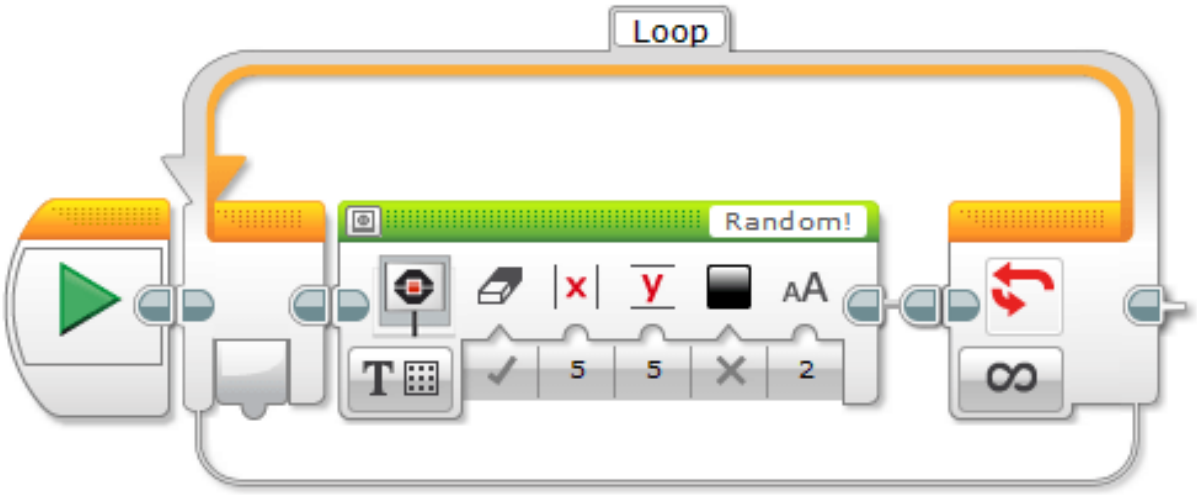



Figura 14.1: Gerador de números aleatórios - Passo 1

Lembra o que o enunciado nos propôs? Para que o Brick gere um número aleatório, é preciso que o botão do meio ( `Center` ) seja pressionado, só dessa forma ele fará o sorteio. Caso outro ou nenhum botão seja pressionado, nada acontece. O bloco `Loop` executa o programa de maneira contínua, não possui mecanismo de tomada de decisões. Por isso, para que o nosso programa se torne mais inteligente e, só gere o número aleatório caso o botão `Center` seja pressionado, vamos utilizar o bloco `Switch` .

Você já deve ter compreendido a função do `Switch` , porém, caso não tenha entendido muito bem, não se preocupe, suas dúvidas serão esclarecidas. Antes de utilizar um bloco, é preciso entender qual é a sua função, como é seu funcionamento. Por isso, vamos ver, a seguir, de forma mais clara e detalhada, qual é a função de um `Switch` .

O `Switch` , assim como o bloco `Loop` , é um bloco de fluxo. Utilizamos o bloco `Switch` quando o programa precisa tomar decisões, fazer escolhas. Por exemplo, se alguém lhe oferece algo e você aceita, esta pessoa vai lhe dar o que ela ofereceu, pois sua resposta foi "SIM". Mas se você recusa, ela não vai lhe dar, pois você disse que não quer, você disse "NÃO". Para cada uma das suas respostas existe uma consequência diferente.

Isso também ocorre na programação. Você pode configurar o seu programa para reagir de maneira particular a cada situação, cada `Case` (ou caso). De forma mais simplificada, podemos dizer que o `Switch` é uma estrutura que permite tomar decisões para múltiplos casos. Dessa forma, podemos predeterminar no `Switch` uma sequência de ações para o programa executar, quando a situação for verdadeira, e outra para quando for falsa. Agora que você já sabe o que é um `Switch` , podemos continuar.

O `Switch` foi configurado em `Brick Buttons` no modo `Compare` , que foi escolhido pelo botão do meio, o `Center` . Isso é realizado na entrada `Set of Brick Buttons ID´s` . A entrada `State` continua com o valor 1, que representa o estado pressionado ( `Pressed` ). Portanto, ele compara se o botão do meio foi pressionado e, caso tenha sido, executa as ações referentes ao estado `True` do bloco `Switch` ; caso contrário, será acionado o estado `False` , que não possui nenhum bloco. Essa ausência de bloco significa que o fluxo do programa permanecerá na ação anterior, que consiste na exibição da palavra “Random!”.

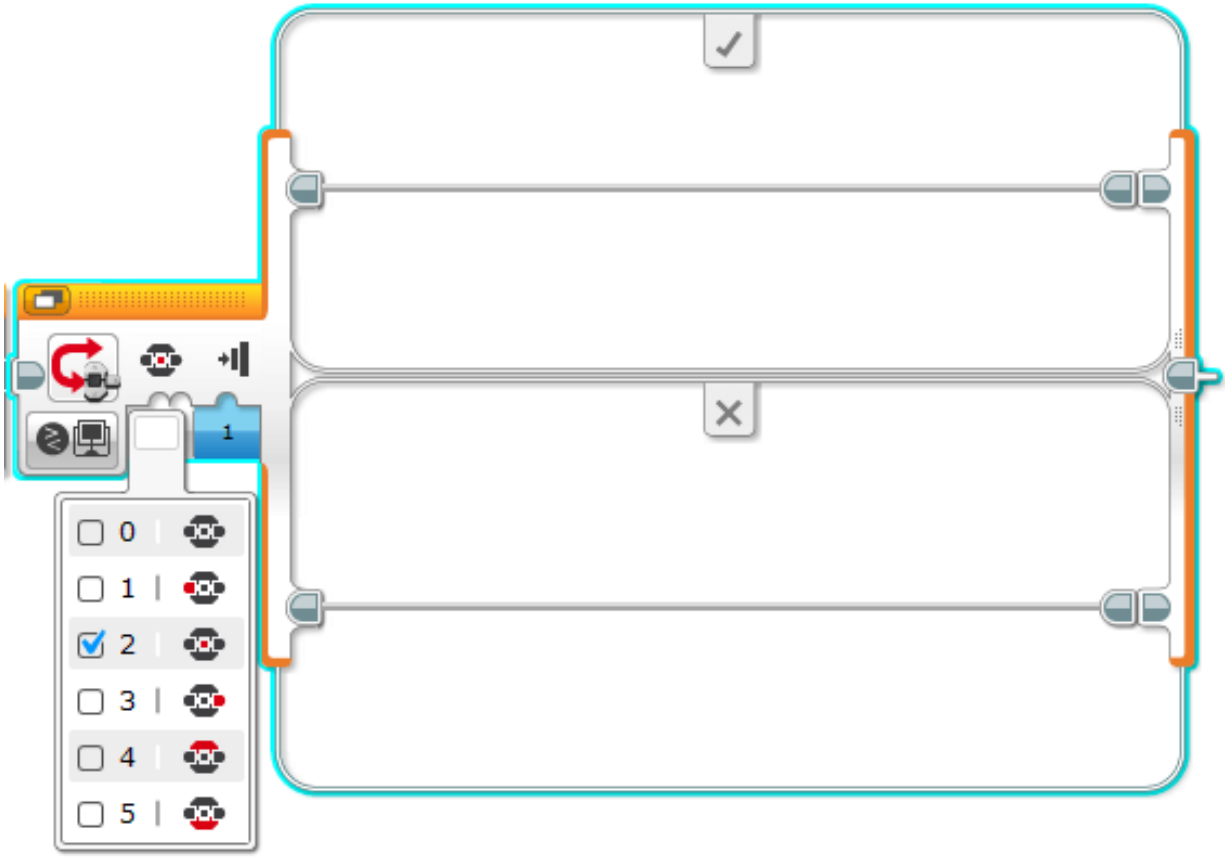



Figura 14.2: Gerador de números aleatórios - Passo 2

No `Switch` , está o bloco `Random` , que gera valores aleatórios. Ele é configurado em `Numeric` para que possa gerar números. Deve-se também configurar a faixa de limite, em que os números serão gerados. Na entrada `Lower Bound` (limite inferior) foi escolhido o número 1. Em `Upper Bound` (limite superior) foi escolhido o número 10. É necessário guardar esse número gerado, para que o bloco `Display` possa utilizá-lo e exibi-lo. Para isso, utilizamos uma variável. Antes de prosseguir com o nosso programa, vamos saber um pouco mais sobre as variáveis, que constituem um recurso importantíssimo recurso.

## 14.1 Variáveis

Variáveis são de grande importância na programação. Tratam-se de posições de memória associadas a identificadores, que armazenam resultados de expressões, obtidos durante a execução de um programa. As variáveis são utilizadas para guardar informações a serem usadas pelos programas. O próprio nome já é sugestivo, indicando que elas têm seu conteúdo modificado de acordo com as necessidades do programa. Uma variável pode permanecer com o mesmo valor durante um programa, mas sua característica é que o valor registrado ali não é imutável.

Em um programa, variáveis são usadas para guardar algo, uma informação, mas também podem ser usadas para “fornecer” informações. Nesse modo as informações contidas nelas serão lidas pelos blocos.

Como criar uma variável em um programa?

Na paleta de blocos clique na aba `Data Operations` , em seguida, no bloco `Variable` .

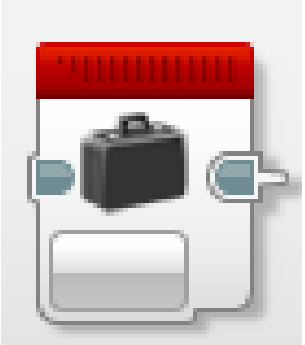


Figura 14.3: Ícone do Bloco Variable

Podemos configurar uma variável para trabalhar em dois modos, são eles: o modo `Read` (ler) e o modo `Write` (escrever). Depois de definido o modo, escolhemos com qual dos tipos de informação a variável trabalhará, que podem ser: `Text` , `Numeric` , `Logic` , `Numeric Array` ou `Logic Array` .

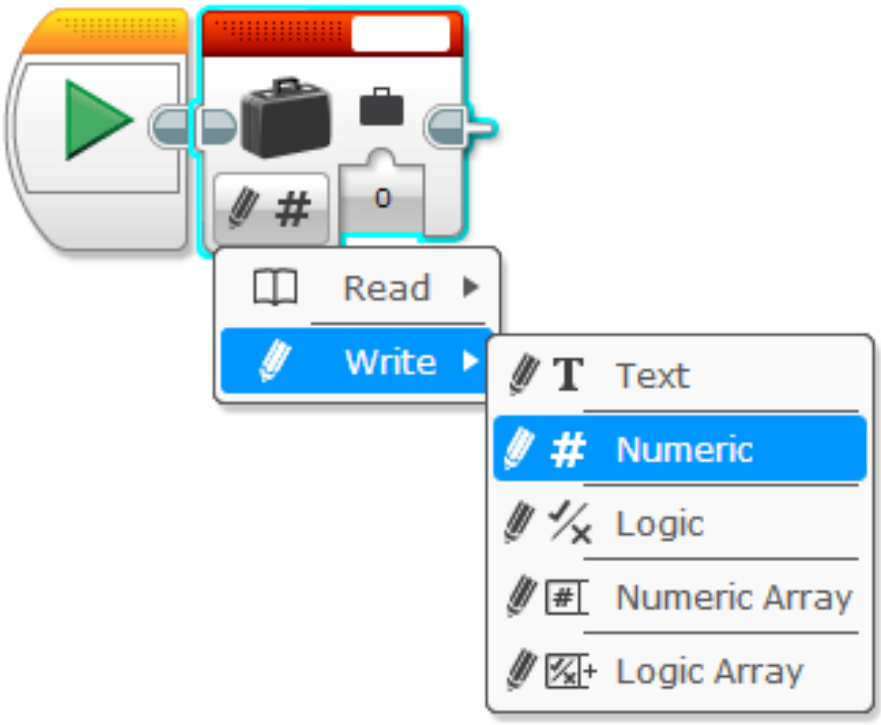



Figura 14.4: Bloco Variable - Modos e tipos de variáveis

Agora que o bloco foi configurado, é necessário adicionar a variável. Fazemos isto clicando no retângulo branco do canto superior direito do bloco, quando aparecerá uma única opção para ser selecionada, identificada pelo botão `Variable Name` , o qual deve ser pressionado pelo mouse.

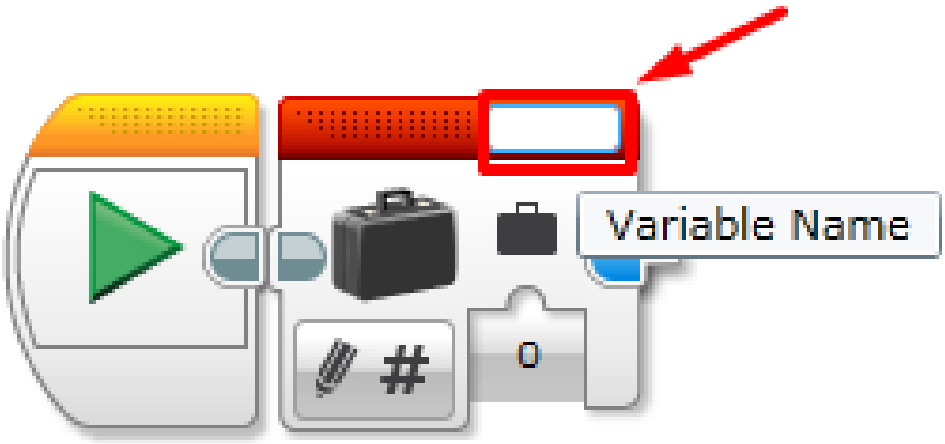



Figura 14.5: Bloco Variable - Adicionando um nome à variável

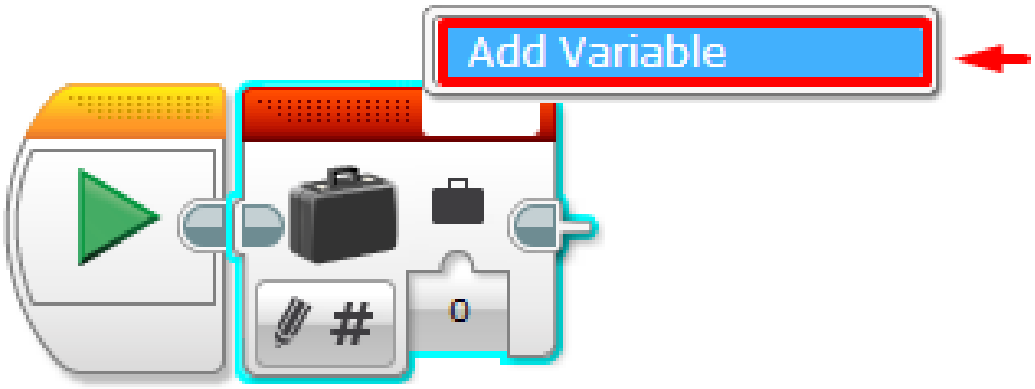



Figura 14.6: Bloco Variable - Adicionando um nome à variável

A seguinte tela será mostrada:

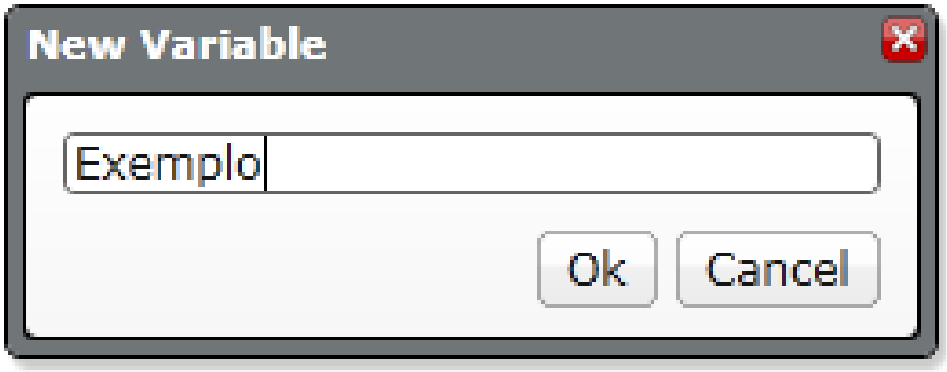



Figura 14.7: Bloco Variable - Adicionando um nome à variável

Dentro da caixa, inserimos um nome para a variável. No caso da imagem, o nome escolhido foi “Exemplo”. É importante salientar que o nome da variável só pode conter os seguintes caracteres:

Figura 14.8: Bloco Variable - Caracteres admitidos na nomeação da variável

Ok, agora você já sabe para que serve, como funciona uma variável e como adicioná-la. Podemos continuar com o nosso programa.

Para guardar o valor gerado pelo bloco `Random` , adiciona-se uma variável no modo `Write` , do tipo `Numeric` . É necessário nomear a variável, conforme vimos há pouco; o nome escolhido neste caso foi `Botton` . Com isso, o número gerado será escrito na variável. Agora, conecta-se a saída `Value` do bloco `Random` à entrada `Value` da variável, o `Variable` . O número gerado já está guardado; o próximo passo é exibi-lo na tela.

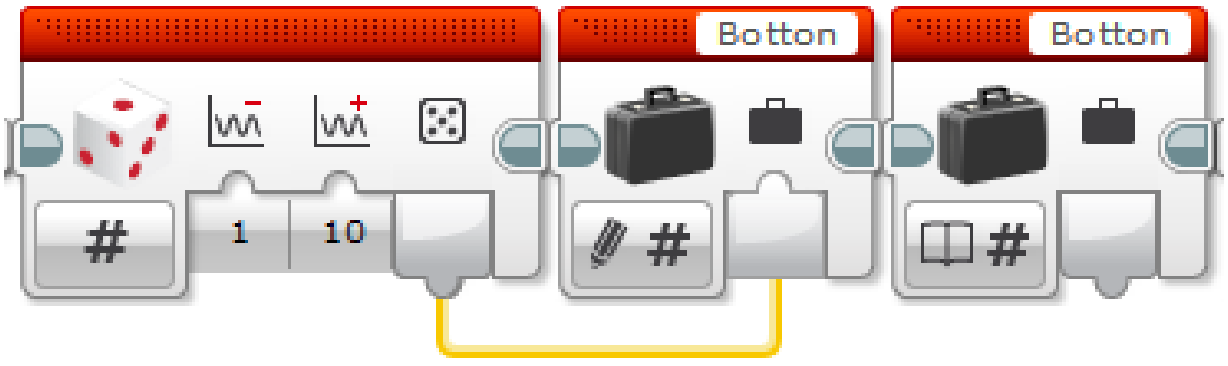



Figura 14.9: Gerador de números aleatórios - Passo 3

Para exibir o número, é necessário que o `Display` leia o valor gerado, que foi guardado na variável `Botton` . Para isso, adiciona-se o bloco

Variable , o qual deve ser configurado para o modo Read (ler), e em seguida Numeric . Agora, adiciona-se o bloco Display . (Neste caso, foi utilizado o modo Text - Pixels , com as seguintes coordenadas: x = 50, y = 75, pois esses valores permitem uma visualização agradável, mas pode-se utilizar o modo Grid , ou mudar os valores das coordenadas para obter um posicionamento de caracteres mais aprazível).

Deve-se mudar o bloco para a opção Wired , situada no canto superior direito do bloco, para que seja exibido o valor enviado por meio do Data Wires (fio de dados). Em seguida, conecta-se o fio da saída Value da variável à entrada Text do Display . Por último, tem-se o bloco Wait configurado em 3 segundos. Esse bloco faz com que o número seja exibido na tela por 3 segundos, e só depois disso o programa retornará ao início.

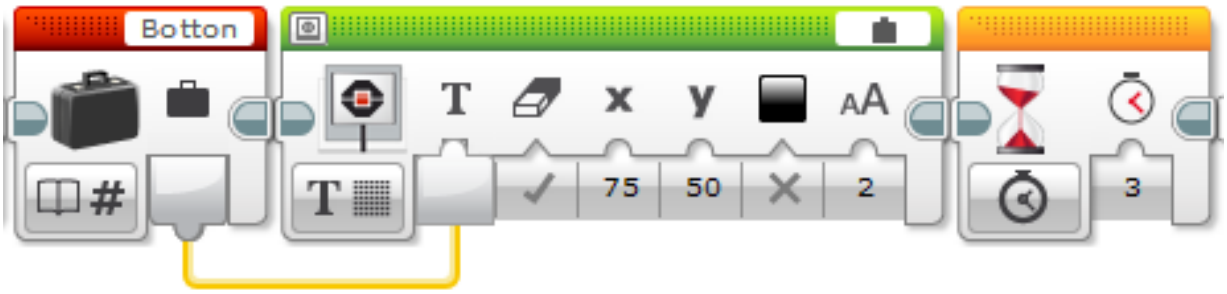



Figura 14.10: Gerador de números aleatórios - Passo 4

O programa ficou da seguinte forma:

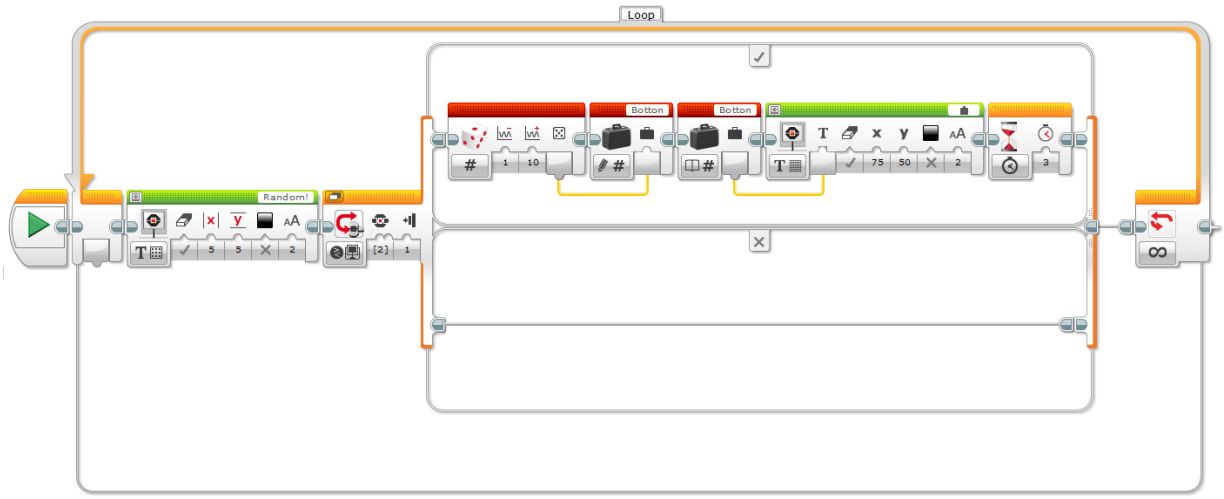



Figura 14.11: Gerador de números aleatórios - Código completo

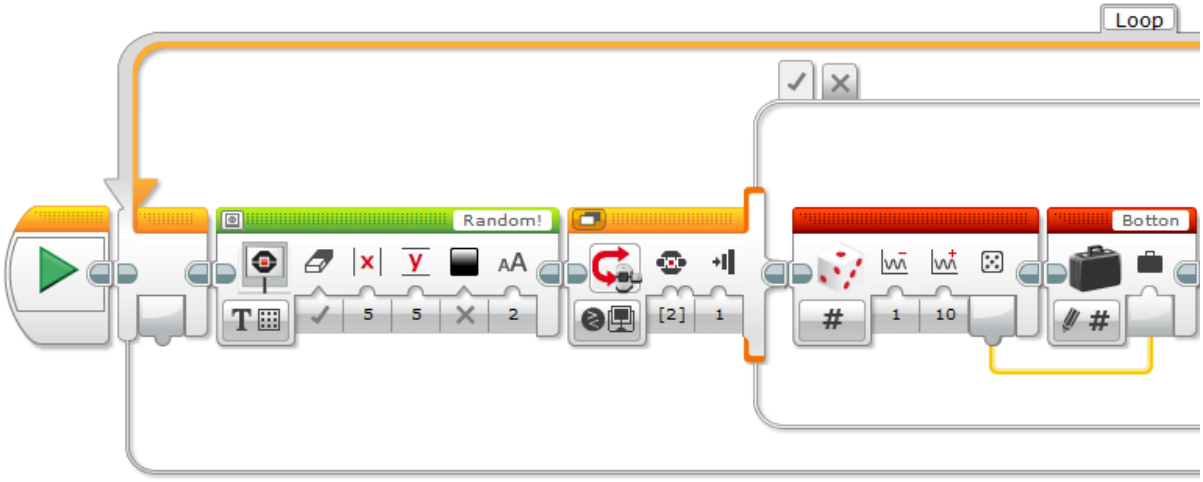



Figura 14.12: Gerador de números aleatórios - Detalhe 1

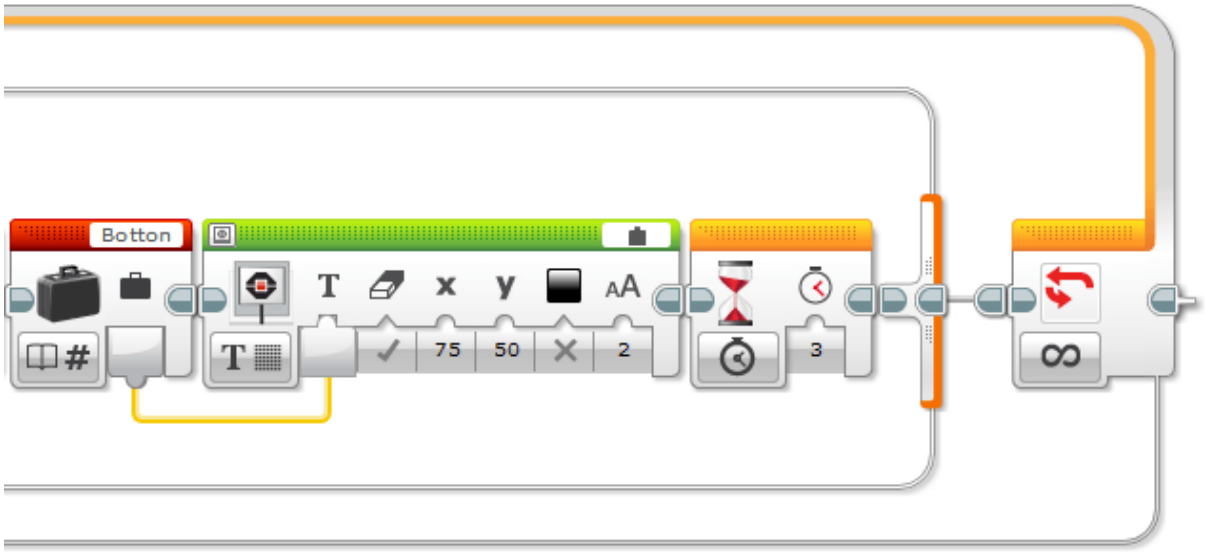



Figura 14.13: Gerador de números aleatórios - Detalhe 2

## 14.2 Modos de exibição do Switch

Também é possível alterar a forma de visualização do `Switch` . O `Switch` apresenta dois modos de visualização: o modo `Flat View` e o `Tabbed View` .

Para isso, basta clicar no `Switch to Tabbed View` , conforme a imagem a seguir. Dessa forma, o `Switch` será exibido no modo `Tabbed View` .

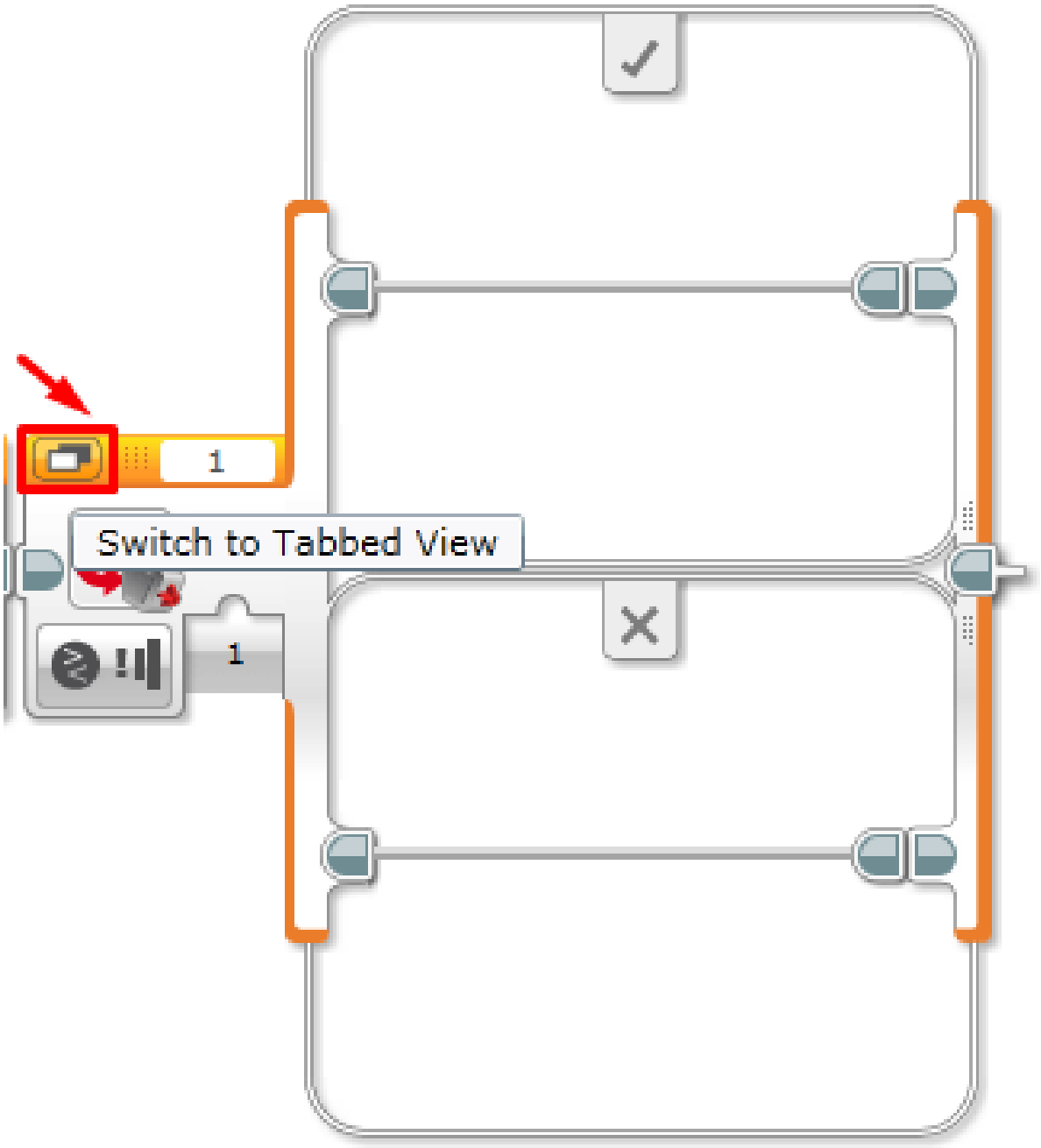



Figura 14.14: Bloco Switch - Switch no modo de exibição Flat View

O `Switch` ficará assim:

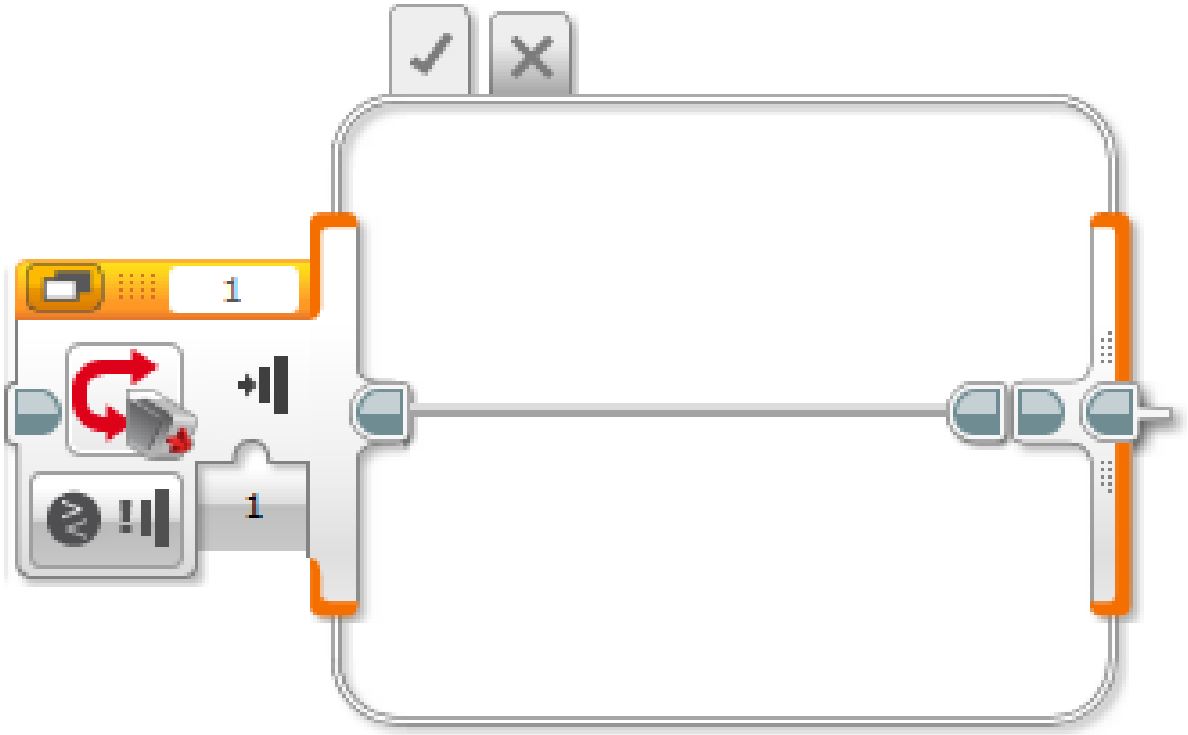



Figura 14.15: Bloco Switch - Switch no modo de exibição Tabbed View

Para voltar ao modo `Flat` , basta clicar no mesmo lugar.

## 14.3 Utilizando variáveis - gerador de números aleatórios com variáveis

Este programa realiza as mesmas funcionalidades do código anteriormente explicado. Porém, apresenta outra lógica. Talvez a resolução anterior seja mais prática e rápida. Entretanto, é uma tarefa interessante o aprendizado de outras formas de programação, por isso também é importante estudar esta nova solução.

No último exemplo, utilizou-se o bloco `Switch` configurado em `Brick Buttons - Compare` , para comparar se o botão tinha sido pressionado. Contudo, pode-se apenas utilizar um bloco `Switch` configurado no

modo `Logic` . Para isso, usa-se o sensor `Brick Buttons` , e o seu resultado é armazenado em uma variável, que transmitirá o valor ao `Switch` .

Pouca coisa muda. Mais precisamente, essas são as alterações:

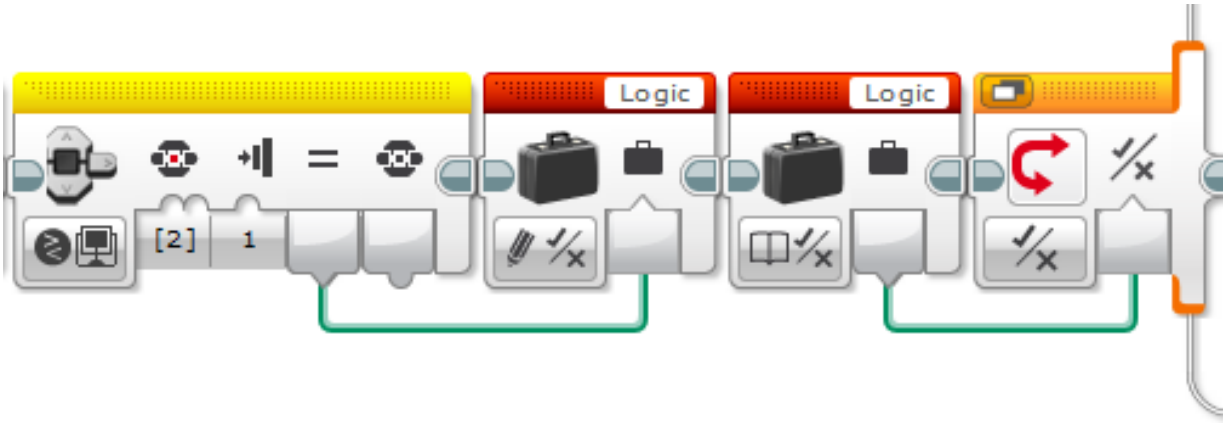



Figura 14.16: Gerador de números aleatórios com variáveis

O primeiro bloco é um bloco sensor. Trata-se do `Brick Buttons` ; com ele, pode-se saber se o botão para o qual está configurado foi pressionado ou não. Para isso, vamos configurá-lo no modo `Compare – Brick Buttons` . Na entrada `Set of Brick Buttons ID´s` escolhe-se o botão que será comparado, que será a opção `Center` . O estado é o mesmo do programa anterior, o `Pressed` .

O próximo bloco é o da variável denominada `Logic` . Ele foi configurado no modo `Write – Logic` . Aqui, utiliza-se o modo `Logic` , pois o `Switch` é do tipo `Logic` . A lógica funciona da seguinte forma: caso o botão comparado tenha sido pressionado, é escrito o valor `True` , caso contrário, o valor `False` será gravado. Por último, este valor é fornecido ao `Switch` por meio da mesma variável, só que configurada no modo `Read` .

Na imagem a seguir, é possível conferir todo o código e observar o resultado final.

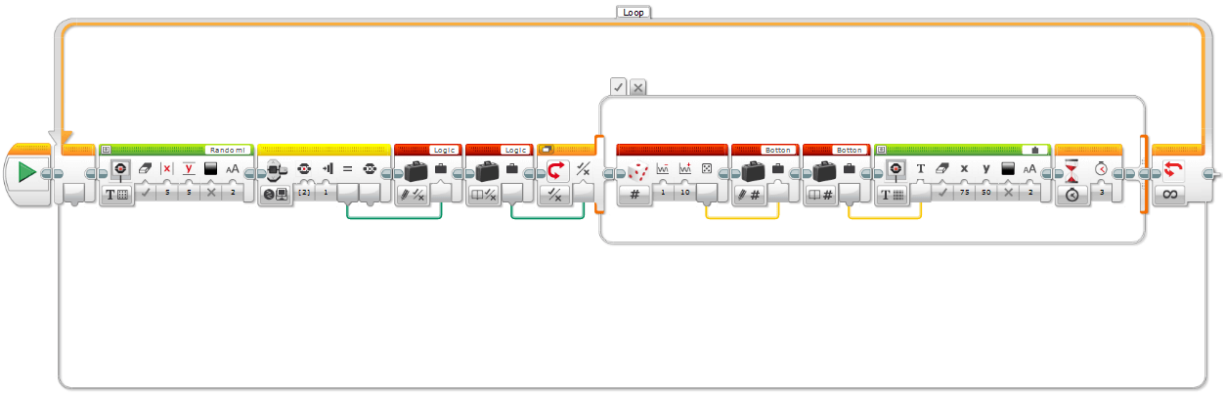



Figura 14.17: Gerador de números aleatórios com variáveis - Código completo

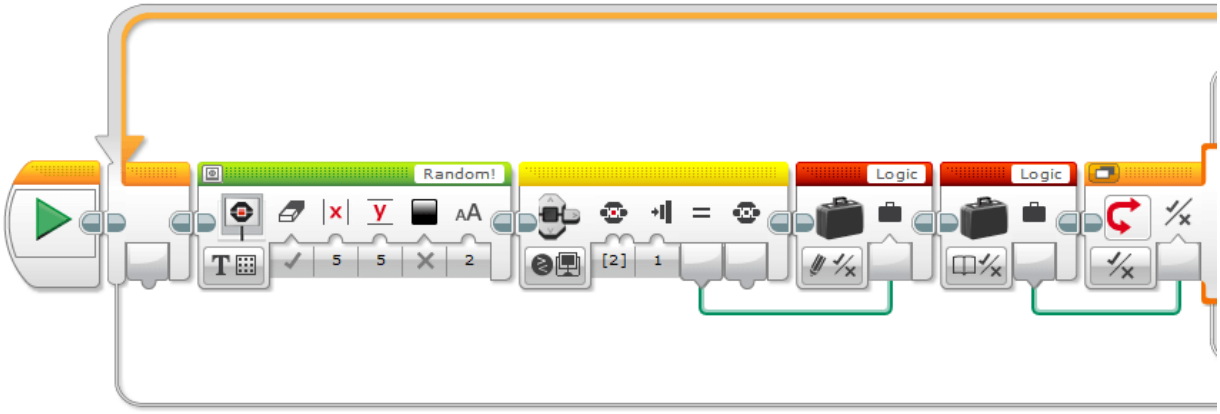



Figura 14.18: Gerador de números aleatórios com variáveis - Detalhe 1

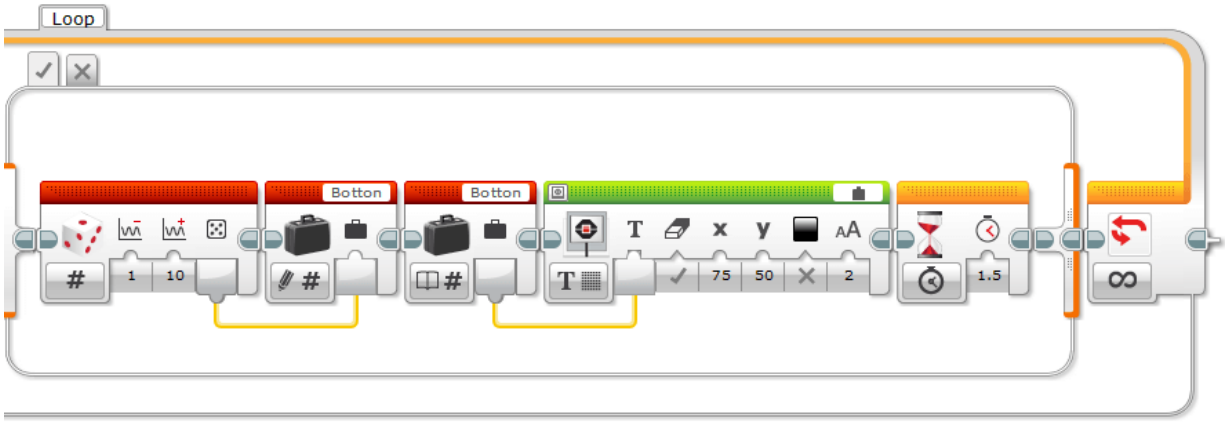



Figura 14.19: Gerador de números aleatórios com variáveis - Detalhe 2

## 14.4 O bloco Random

No programa que acabamos de fazer, utilizamos o bloco `Random` para gerar números aleatórios. Porém, o bloco `Random` não faz apenas isso, ele faz um pouco mais. O bloco `Random` gera **valores aleatórios**. Isso quer dizer que podemos utilizá-lo para gerar um valor numérico (modo `Number` ) ou um valor lógico (modo `Logic` ): `True` ou `False` .

No nosso programa, utilizamos um limite entre 1 e 10, mas você pode utilizar limites muito maiores, pode até mesmo utilizar números negativos. Contudo, no modo `Logic` você só pode utilizar um limite de 0 a 100, pois este limite representa a probabilidade, que varia de 0% a 100%.

## 14.5 Tente fazer

Conforme dito anteriormente, é muito importante que você pratique. E para isso, não é necessário mudar todo o código, ou criar um novo programa. A melhor forma de descobrir e sanar suas dúvidas é testando seu código. Desse jeito, você consegue ver por conta própria os resultados obtidos. Para finalizar este capítulo, teste as seguintes alterações:

- O bloco `Random` , pode ser utilizado com limites muito superiores que o limite 1 e 10. Por isso, altere o valor do limite, coloque, por exemplo, entre 1 e 100, 1 e 1000.
- Além de números positivos, podemos utilizar os números negativos. Modifique o limite do seu bloco `Random` utilizando números negativos.
- E para finalizar, tente exibir na tela do seu Brick um valor lógico ( `True` ou `False` ). Vale lembrar que, para utilizar o modo `Logic` , é preciso fazer mais algumas alterações no programa. Não basta

apenas selecionar o modo `Logic` e definir a probabilidade desejada. Seria necessário modificar o tipo da nossa variável. Para armazenar um valor lógico, é preciso utilizar o modo `Logic` . Mas não se esqueça de mudar o tipo das duas variáveis, tanto a de escrita (que está no modo `Write` ), quanto a de leitura (que está no modo `Read` ).

Mais um capítulo concluído, agora você já pode explorar os outros modos dos blocos vistos aqui. Você já pode testar todas as possibilidades oferecidas pelo bloco `Random` .

# Capítulo 15
# Identificador de cores

O sensor de cor pode reconhecer sete cores: verde, amarelo, azul, preto, marrom, vermelho e branco. Neste capítulo, vamos criar um programa que sabe dizer qual é a cor e que reconhece, ainda, o modo `No colour` (sem cor).

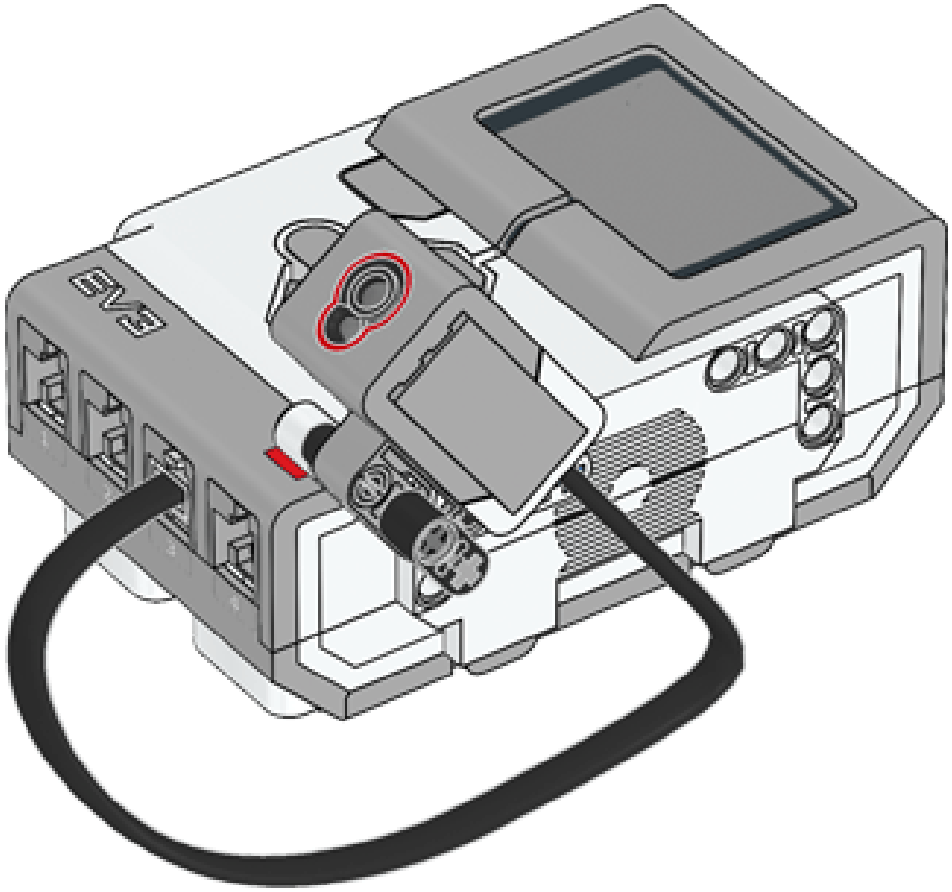


Figura 15.1: Identificador de cores

## Instruções de montagem

Este projeto requer apenas o Brick e um sensor de cor conectado a ele. Mas, caso surja alguma dúvida quanto a montagem, e a edição utilizada seja a Education, basta acessar o *Lobby*, localizar a opção

Building Instructions , em seguida, clicar em Building Ideas e procurar pelo Colour Sensor - Brick . Clique nele e, por fim, clique em Open para abrir o manual.

Para os usuários que não atualizaram o software ainda, basta acessar o *Lobby* e clicar em Robot Educator , Building Instructions , Colour Sensor – Brick e, finalmente, em Open .

## Objetivo do programa

O programa primeiramente exibe a mensagem “Go!” na tela (display) do Brick. Quando detectado que o botão do meio foi pressionado, o sensor será acionado e fará a leitura da cor. O resultado obtido é exibido na tela por 3 segundos.

## O código

O primeiro bloco após o bloco Start é um Loop – Unlimited . Dentro deste Loop está o bloco Display , que exibe a mensagem “Go!”. Foi usado o modo Text – Grid , com as seguintes coordenadas: x = 2, y = 2, porque eles oferecem um bom posicionamento para a mensagem que vamos exibir. Mas, conforme já foi visto, você pode alterar esses valores para os que mais lhe agradarem.

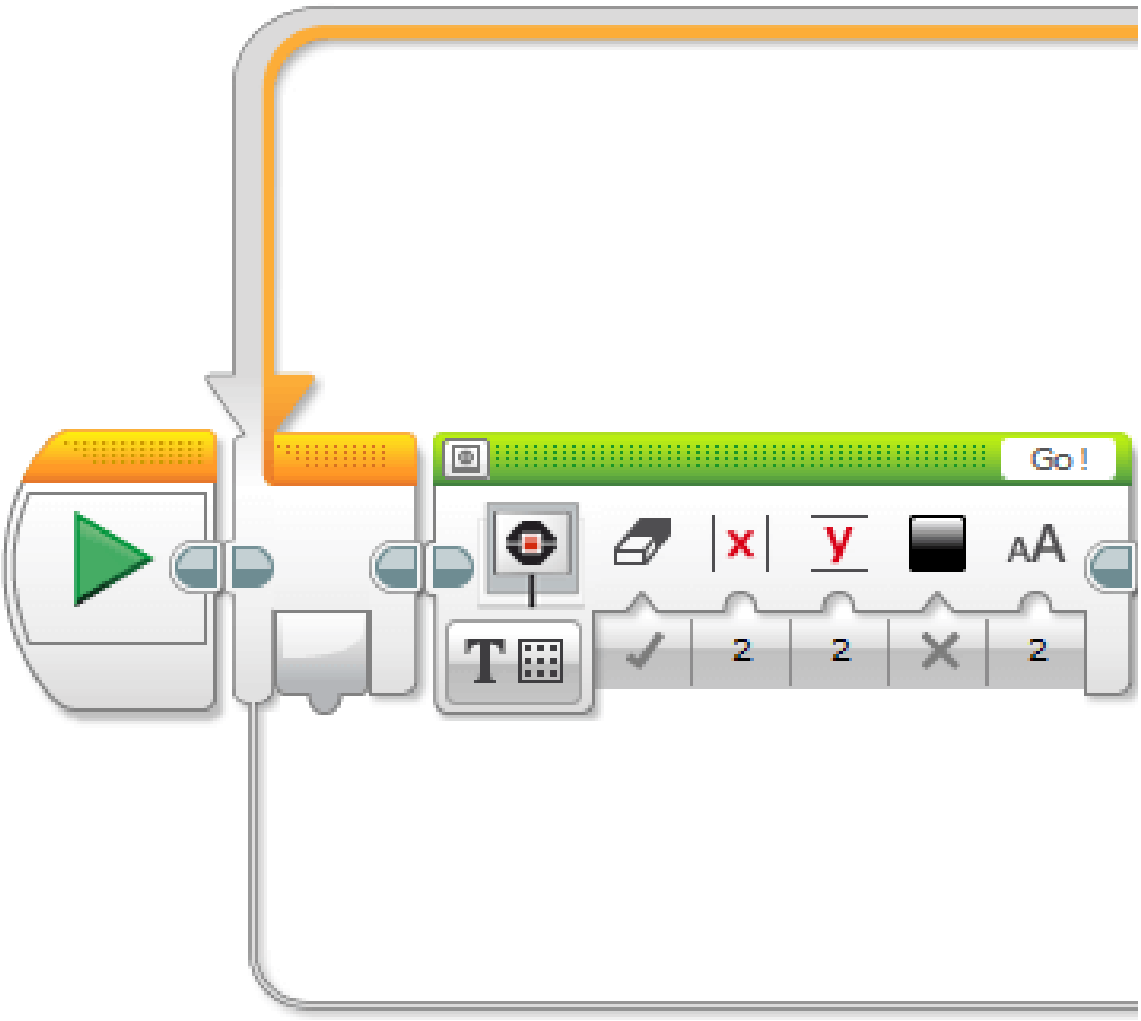



Figura 15.2: Identificador de cores - Passo 1

Logo após, há um bloco `Switch` , que executa a mesma função dos programas explicados anteriormente: ele compara se o botão do meio foi pressionado. Dentro do `Switch` está o bloco sensor `Colour Sensor` , responsável por fazer a leitura da cor. O bloco foi configurado no modo `Measure – Colour` , para poder identificar a cor lida.

Depois de feita a leitura, é necessário guardar o valor lido em uma variável.

Cada uma das sete cores possui um número que a representa, (0 - sem cor, 1 - preto, 2 - azul, 3 - verde, 4 - amarelo, 5 - vermelho, 6 -

branco e 7 - marrom) e é este número que será retornado pelo sensor. Isto é, após o sensor identificar a cor, ele envia o número referente a ela na variável `Cor Lida` , configurada em `Write - Numeric` . Para isso conecta-se a saída `Colour` do sensor à entrada `Value` da variável.

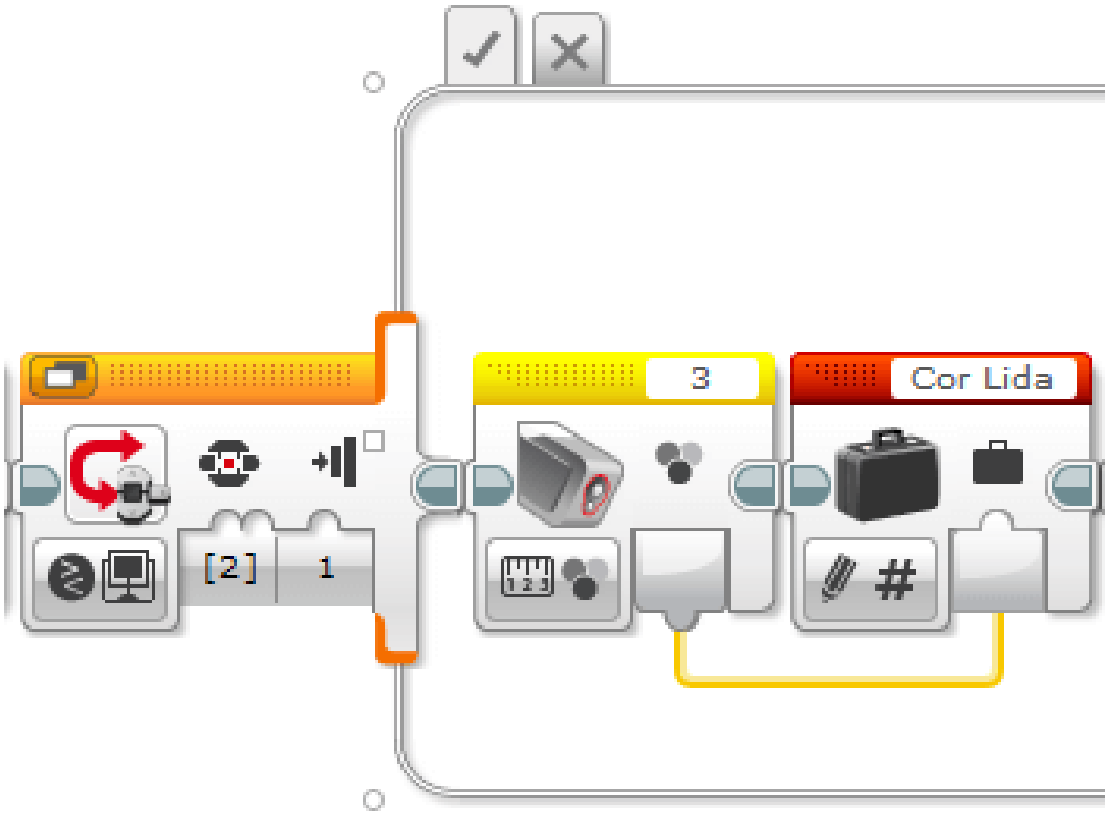



Figura 15.3: Identificador de cores - Passo 2

Agora vamos fazer com que a cor lida pelo sensor seja exibida na tela. O primeiro passo é adicionar a variável `Cor Lida` , configurada no modo `Read - Numeric` .

Exibir esses números que representam as cores na tela não é muito informativo, pois uma pessoa que não sabe que cada cor possui um valor não entenderá nada do que está acontecendo. Podemos deixar o nosso programa muito mais completo, podemos exibir o nome da cor na tela. Mas, como fazer isso?

Vamos utilizar um `Switch - Numeric`. A saída `Value` da variável é conectada à entrada `Number` do `Switch`. Utilizaremos o modo `Numeric`, pois, para cada número retornado pelo sensor informaremos sua cor correspondente (em forma de texto).

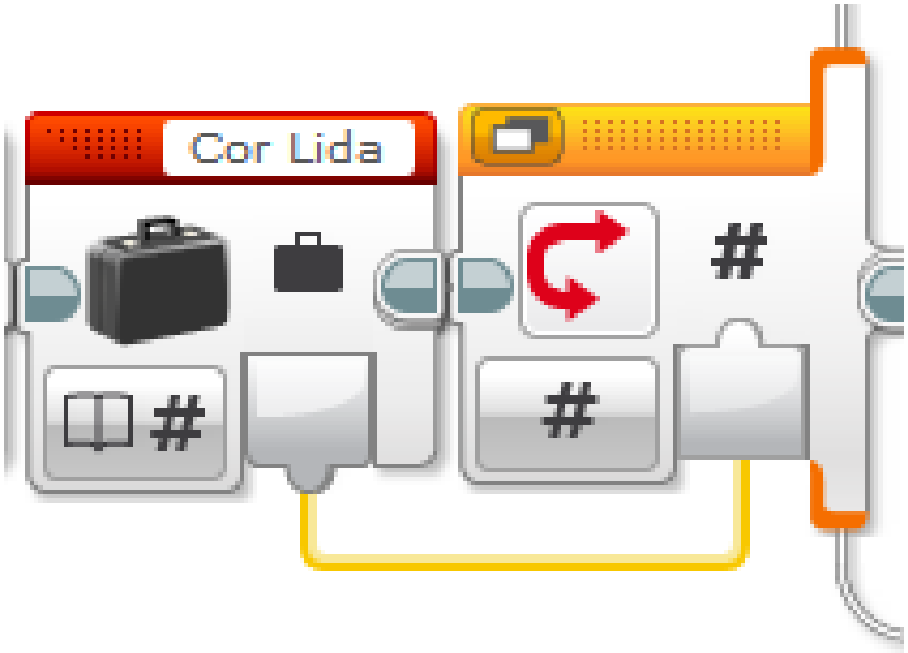



Figura 15.4: Identificador de cores - Passo 3

Porém, ainda falta uma coisa. Ao todo, temos 7 cores mais o modo sem cor. Logo, esses dois `Cases` do `Switch` não serão suficientes. Precisamos adicionar mais `Cases`.

## Adicionando Cases ao Switch

Para fazer isso, devemos criar os casos necessários `Cases` (ou casos) no `Switch`. Nós já vimos que a função do `Switch` é oferecer ao programa possibilidades de escolha. Até agora só trabalhamos com `Switch` no modo `Logic`. Como ele estava no modo `Logic`, só possuía dois casos: `True` e `False`. Entretanto, estamos utilizando o `Switch` no modo `Numeric`, logo, podemos atribuir mais que dois casos ao nosso `Switch`.

Portanto, devemos adicionar 8 casos no bloco `Switch` . No `Case` “0” adicionamos um `Display` . Vamos utilizar esse `Display` para que o nome da cor referente ao valor seja exibido na tela do nosso Brick. Isso quer dizer que, conforme o número retornado, devemos exibir na tela o nome da cor correspondente.

Para adicionar `Cases` ao `Switch` , basta clicar em `Add case` (representado pelo  , logo após o último `Case` ). Ao adicionar o caso, você pode escolher a cor.

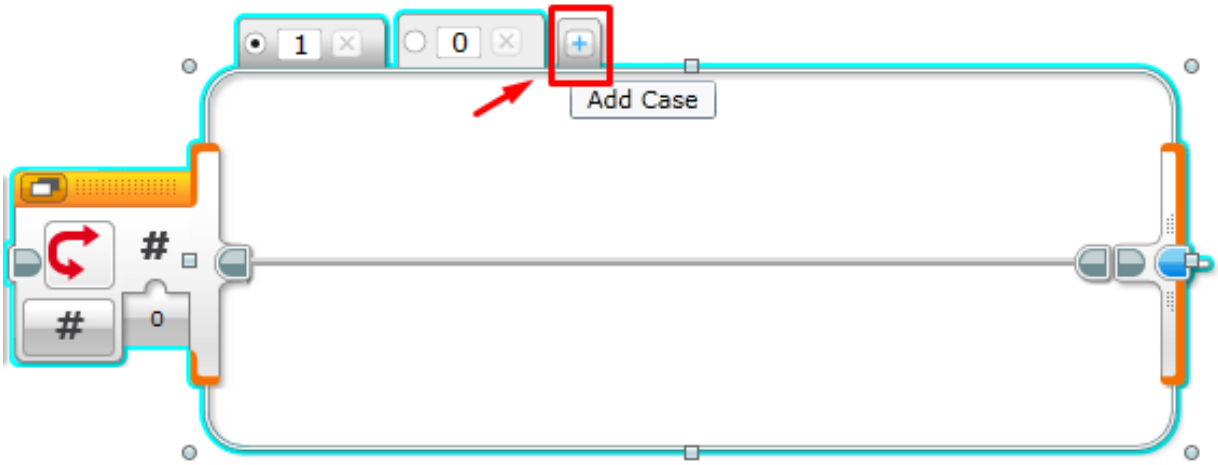



Figura 15.5: Adicionando casos ao Switch

Após adicionar o `Case` , é só nomeá-lo. Precisamos fazer isso para os demais `Cases` . Note que o primeiro `Case` do `Switch` é o "1", logo em seguida, temos o "0". Mas, para facilitar e deixar o programa mais organizado, você pode alterar o número deles, colocando-os em ordem, primeiro o `Case` "0", e em seguida, o `Case` "1".

Certo, agora você já sabe adicionar os `Cases` , então podemos prosseguir. Os passos restantes são muito simples. Para exibir o nome da cor, vamos utilizar o bloco `Display` e escrever nele o nome da cor correspondente ao caso. Observe na imagem a seguir como ficou.

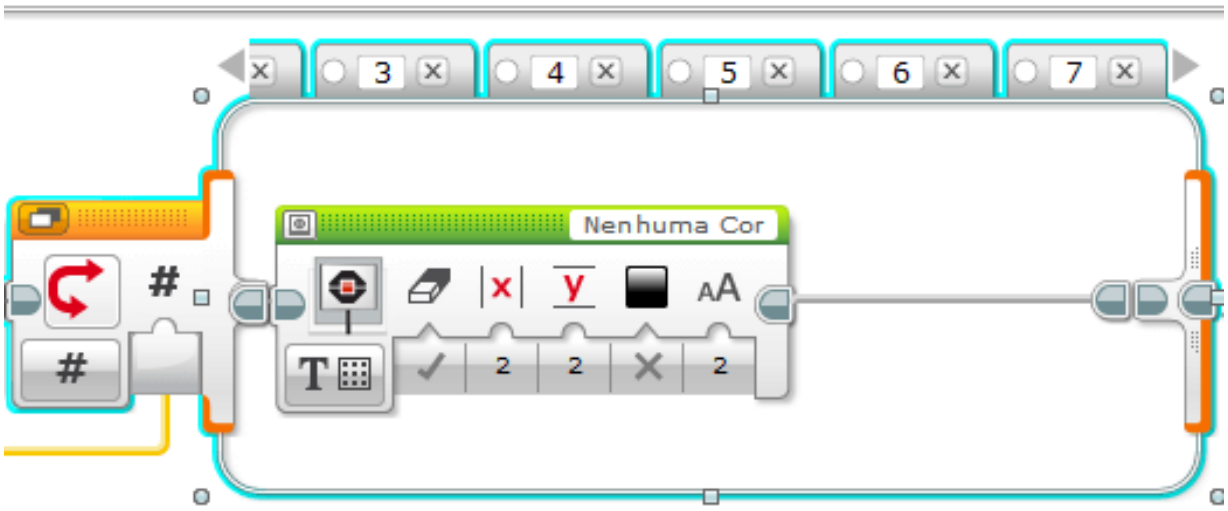



Figura 15.6: Identificador de cores - Passo 4

Faça o mesmo para as demais `Cases` . Por último, vamos adicionar o bloco `Wait` , configurado em 3 segundos, para que o nome da cor lida seja exibida durante 3 segundos.

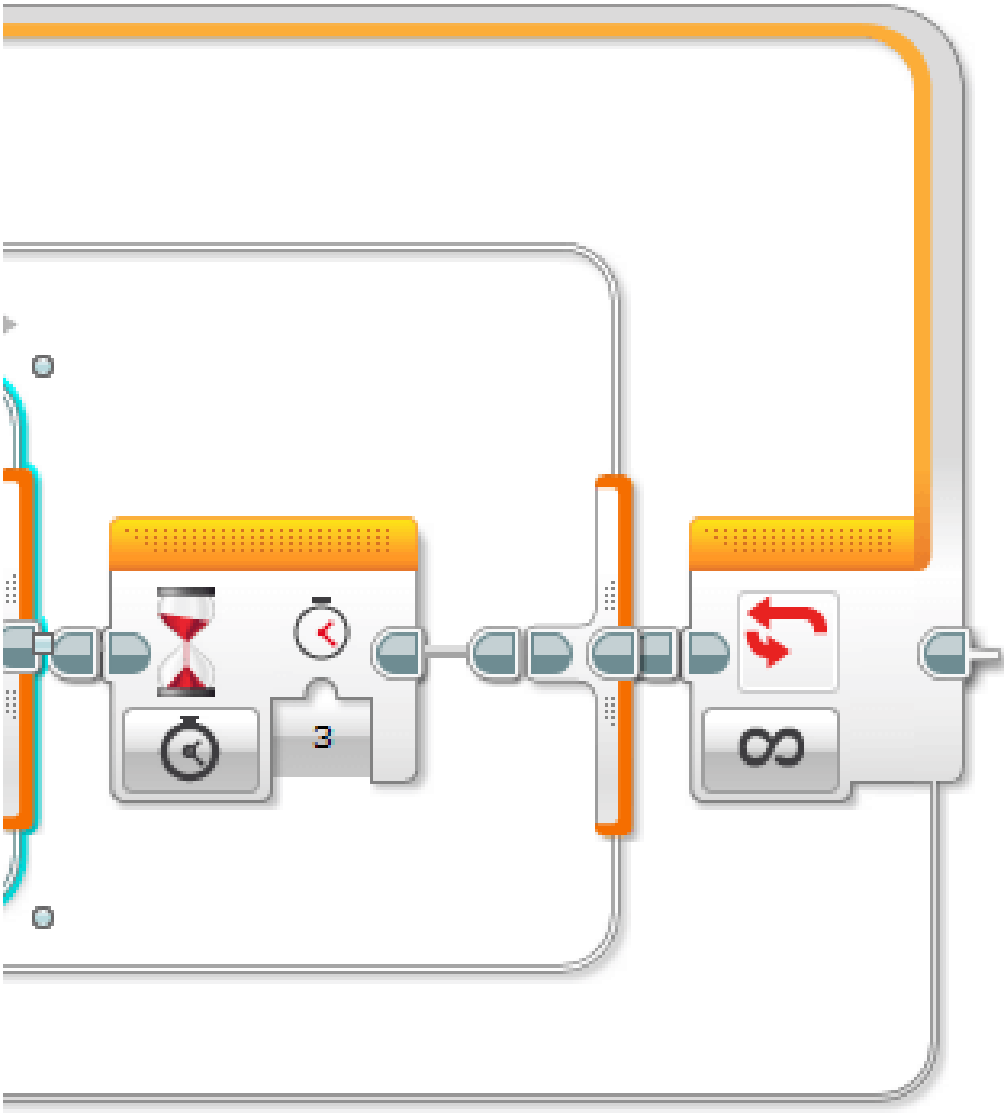



Figura 15.7: Identificador de cores - Passo 5

O programa ficou da seguinte forma:

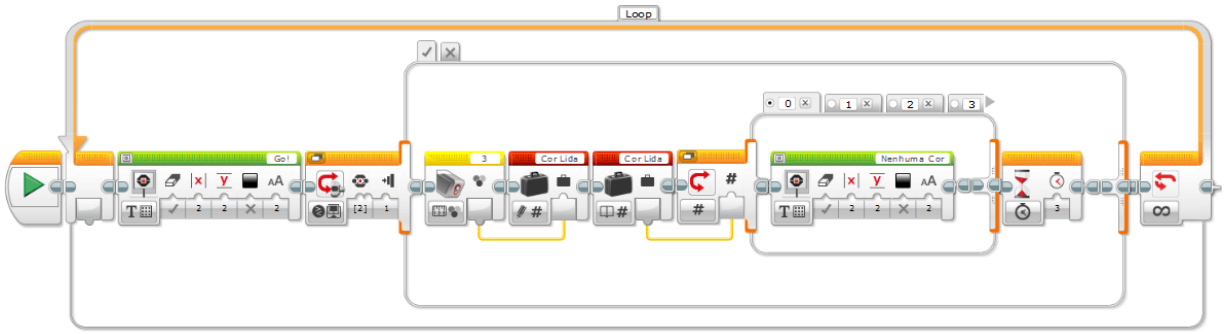



Figura 15.8: Identificador de cores - Código completo

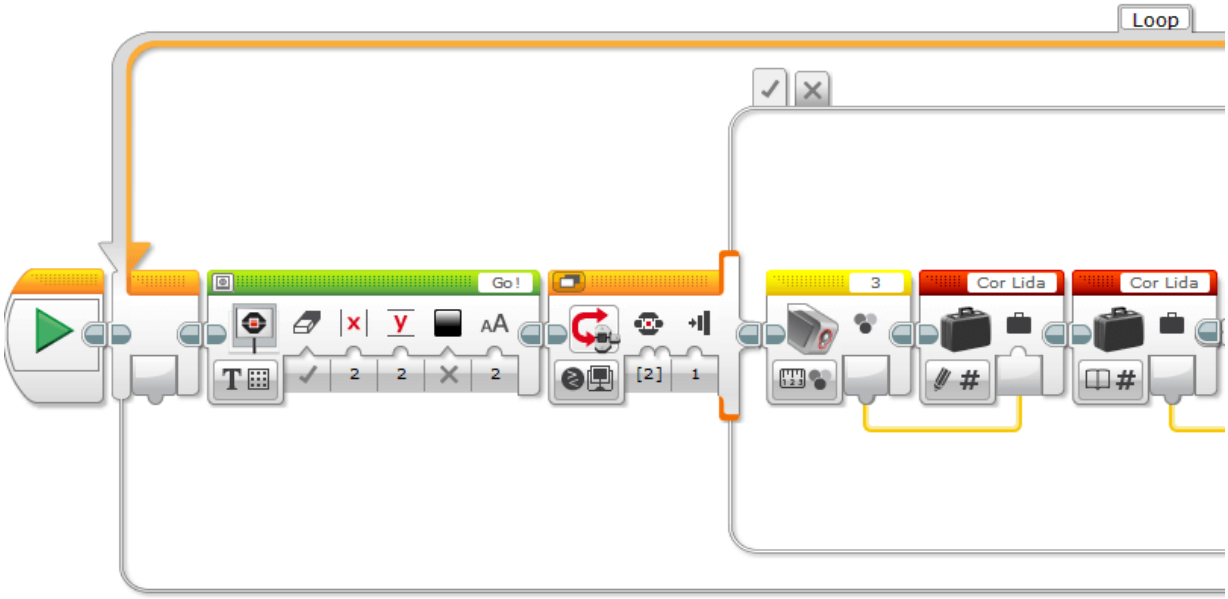



Figura 15.9: Identificador de cores - Detalhe 1

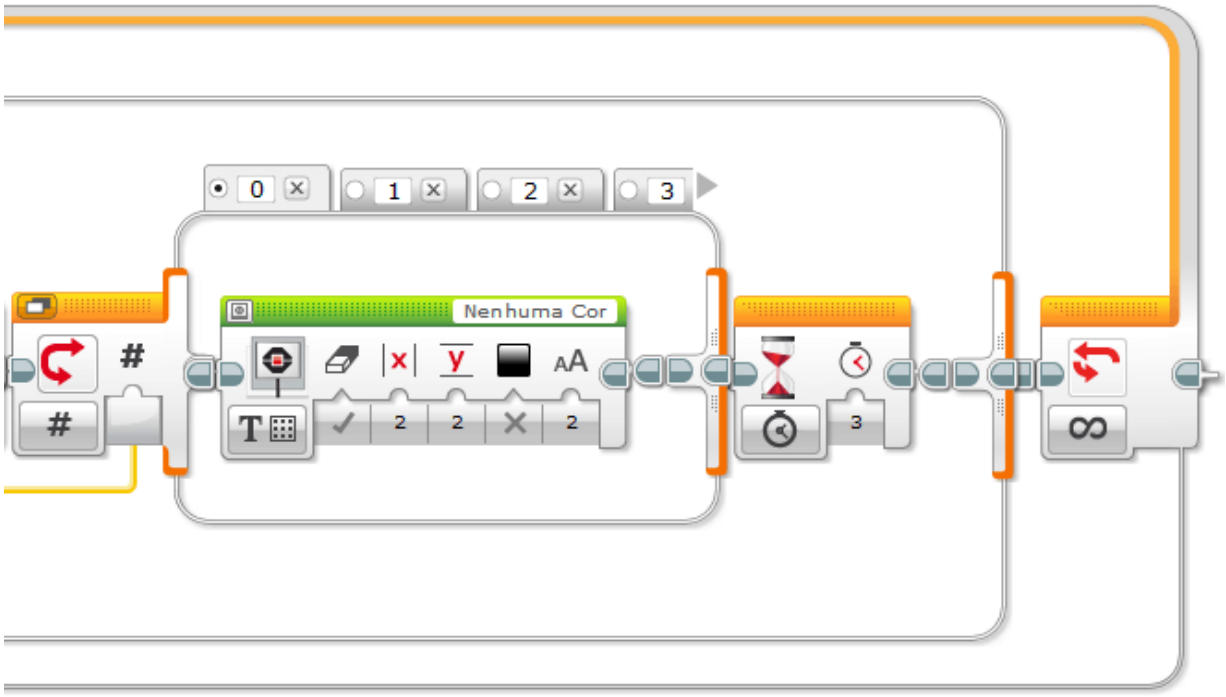



Figura 15.10: Identificador de cores - Detalhe 2

Mais um programa concluído, porém, a seguir vamos aprender uma forma muito mais simples de realizar tudo isso.

## 15.1 Utilizando o Switch no Modo Colour - Identificador de cores com Switch-Colour

No programa anterior utilizávamos o sensor de cor para medir a cor que seria lida (identificando seu número), logo após isso, guardávamos esse valor em uma variável e a fornecíamos para o `Switch` , que de acordo com o número de entrada prosseguia realizando a ação correspondente ao caso.

Contudo, existe uma maneira muito mais de fácil de fazer esse programa. Já vimos que podemos configurar um `Switch` em diversos modos. Um deles é o `Colour Sensor` . Fazendo isso, não precisamos

de um bloco `Colour Sensor` para ler e nem de variáveis para guardar o valor. Podemos utilizar o `Switch` no modo `Colour Sensor` e configurá-lo para medir, e depois é só adicionar os casos. Para isso, basta utilizar o bloco `Switch` com as configurações `Colour Sensor – Measure – Colour` .

Você deve ter reparado que, ao adicionar o `Switch` nesse modo, uma coisa mudou em relação aos `Cases` . O `Switch` agora apresenta não mais `Cases` numéricos, mas sim, `Cases` representados por cores. Sendo assim, ao ler a cor o `Switch` já sabe qual é o `Case` correspondente à cor lida. Vamos entender isso melhor.

Por exemplo, caso a cor lida seja a cor azul, serão realizadas as ações que estão dentro do `Case` referente à cor azul. Se a cor lida fosse a cor vermelha, o programa executaria as ações para o `Case` referente à cor vermelha, e assim por diante.

Agora só devemos adicionar os cases restantes. No primeiro programa, vimos como fazer isso, aqui devemos proceder da mesma forma, mas com um pequeno detalhe. No modo `Numeric` atribuíamos números para designar os `Cases` . Já nesse modo, basta escolhermos a cor para o `Case` . Para fazer isso, clique no símbolo de interrogação. Após isso uma caixa contendo as cores será exibida.

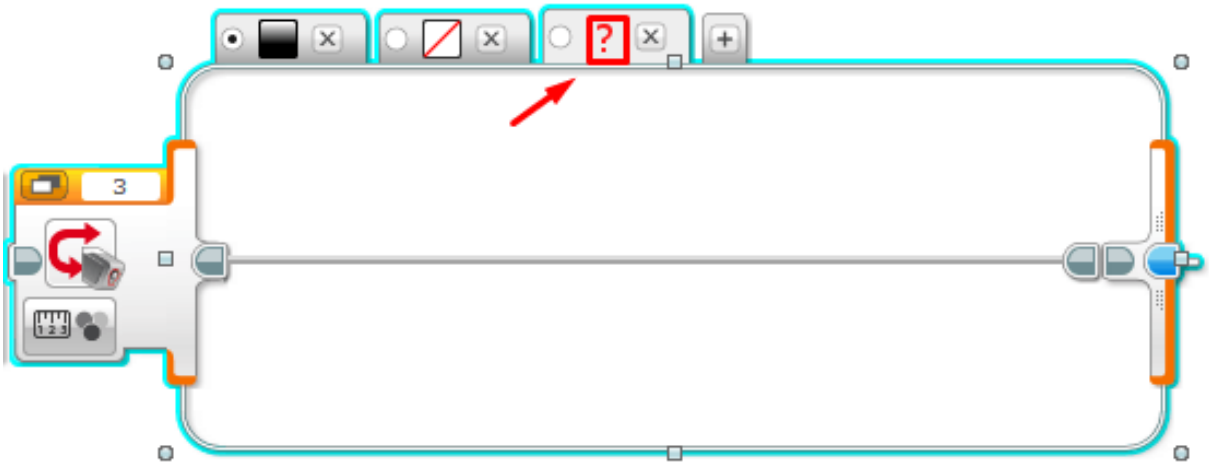



Figura 15.11: Adicionando um Case

O programa ficou da seguinte forma:

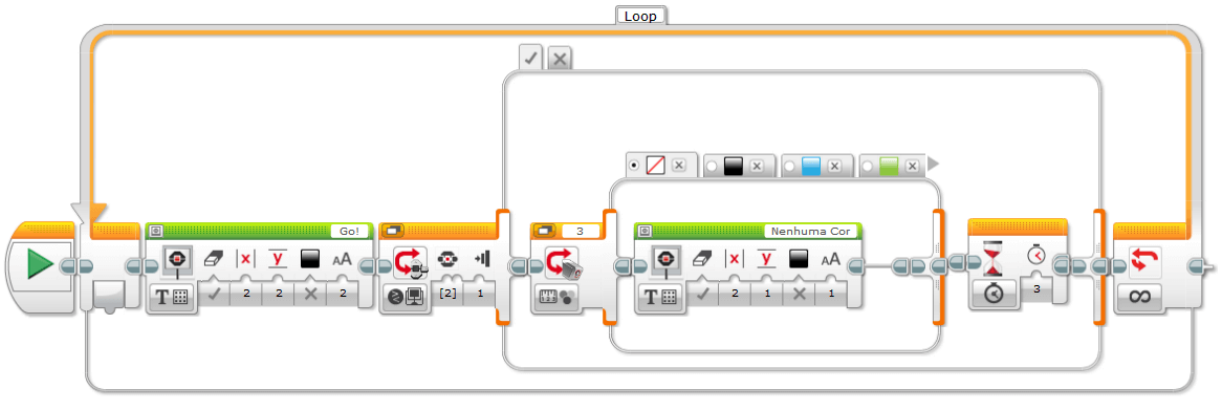



Figura 15.12: Identificador de cores com Switch-Colour - Código completo

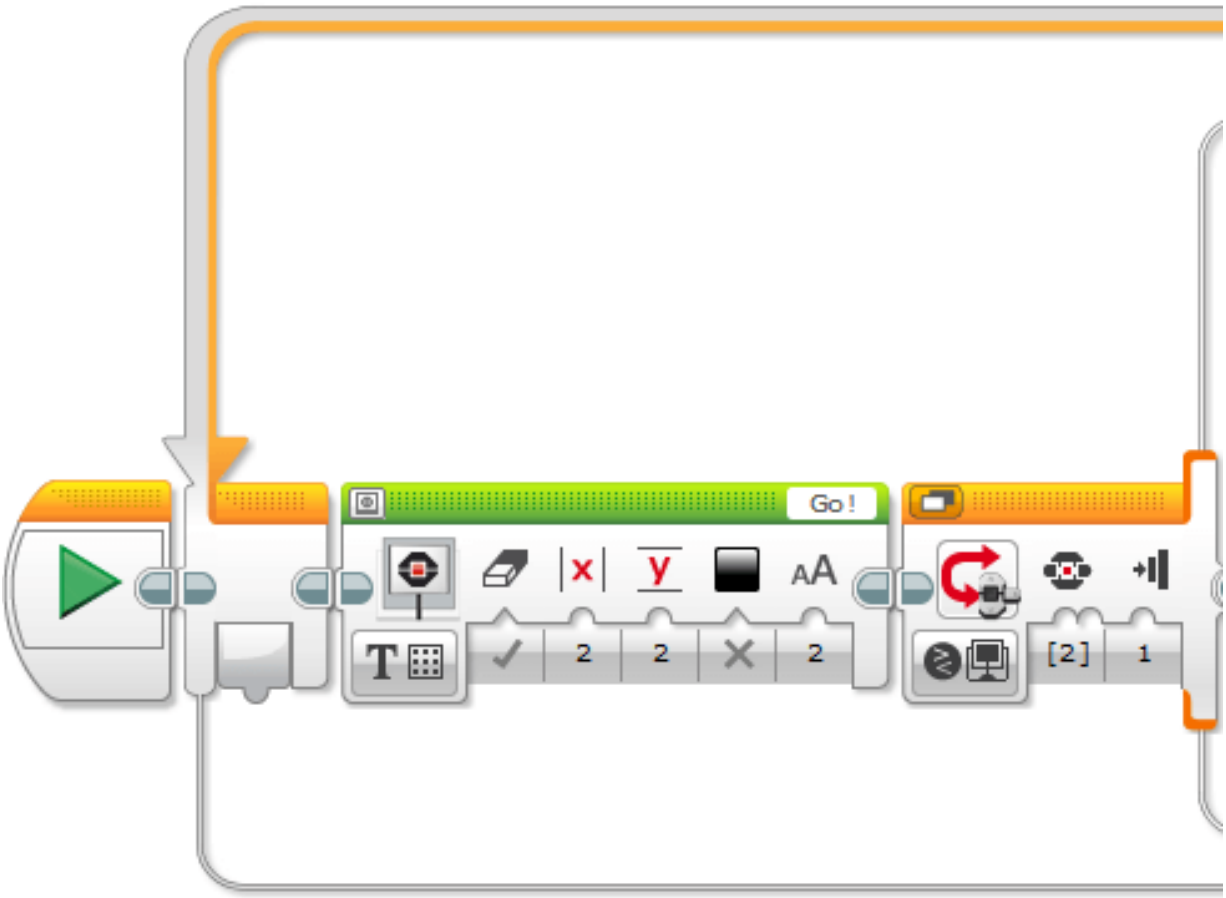



Figura 15.13: Identificador de cores com Switch-Colour - Detalhe 1

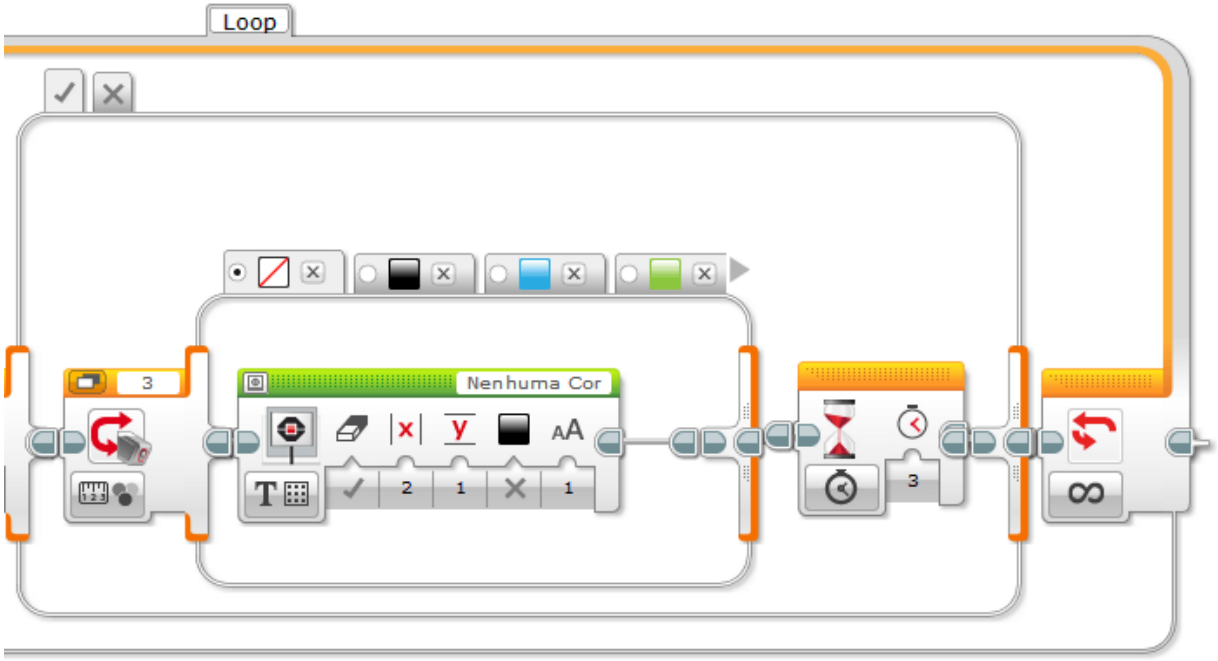



Figura 15.14: Identificador de cores com Switch-Colour - Detalhe 2

Muito mais fácil, não é mesmo?! Nesse momento, você deve estar pensando: "Então por que motivos eu preciso saber fazer o primeiro código, se eu posso fazer tudo de forma automática?". Pois bem, no primeiro código você pôde desenvolver mais as suas habilidades, pois precisava criar uma lógica capaz de resolver o problema, sem utilizar recursos prontos. Assim você treina mais e adquire cada vez mais conhecimentos e experiência.

Agora que você já sabe as duas formas, você pode escolher a que mais gostar. A segunda forma é muito mais fácil e poupa muito tempo. Além disso, ela também reduz drasticamente o tamanho do programa (em relação ao número de blocos).

## 15.2 O bloco Colour Sensor

Este é o bloco responsável pelo sensor de cor, que funciona como sensor de cor ou luminosidade. Nos programas feitos, já o

utilizamos no modo cor, para que realizasse medições. Confira os modos que ele possui:

- *Measure* - Utilizado para realizar a medição em um dos três modos de operação do sensor (modo que utilizamos).
- *Compare* - Utilizado para comparar o valor lido de acordo com um parâmetro estabelecido no bloco. Este modo também possui os três modos de operação citados acima.
- *Calibrate* - É usado para a calibração do sensor (apenas para o modo *Reflected Light Intensity*). Mais à frente, em uma seção própria, será explicada a sua aplicação prática. No decorrer do livro, vamos utilizar todos esses modos. Você ainda aprenderá mais detalhadamente a função do modo `Calibrate` e verá mais a fundo a importância de uma calibração.

## 15.3 Tente fazer

Nos programas anteriores, utilizamos um `Switch - Brick Buttons` e o configuramos para comparar. Escolhemos como referência o botão `Center` , assim quando o botão `Center` é pressionado o programa realiza a identificação, caso contrário, ele permanece exibindo a mensagem "Go!".

Tente fazer com que o seu programa realize a leitura quando qualquer botão do Brick seja pressionado. Após fazer isso, mude a mensagem exibida para "Press any button!", pois o programa fará a identificação da cor assim que qualquer botão for pressionado.

Mude o estado dos botões comparados também, até aqui só utilizamos o `Pressed` , mas não fique preso apenas a esse modo. Mude o modo do `Switch` para o modo `Released` .

No modo `Released` , você deve ter notado que a mensagem "Go!" não foi exibida. Isso ocorre porque, quando selecionamos o modo `Released` (botão solto) e nenhum botão estiver sendo pressionado, a condição para que o `Switch` entre em ação é satisfeita. E, como não colocamos um bloco de espera ( `Wait` ), a mensagem não chega a ser exibida na tela. Para solucionar isso, coloque o bloco `Wait` no modo `Time Indicator` .

# Capítulo 16
# Medidor de luz refletida

No capítulo anterior, utilizamos o sensor Colour Sensor no modo cor. Já vimos que o Colour Sensor do EV3 integra dois sensores em um. Podemos utilizá-lo tanto no modo `Cor` , quanto no modo `Luminosidade` (que pode ser a luz refletida pelo sensor ou a luz do ambiente). Neste capítulo, vamos aprender a utilizar o sensor no modo de `Luminosidade` , mais especificamente no modo `Reflected Intensity Light` , que mede a quantidade de luz refletida.

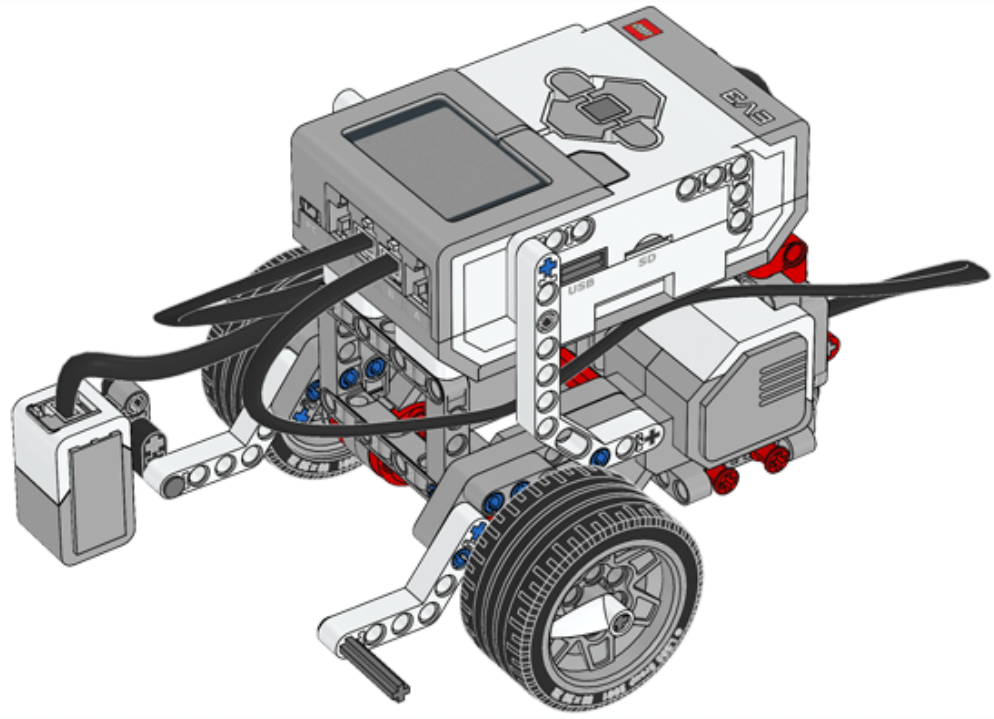


Figura 16.1: Medidor de luz refletida

Este programa é muito simples. Vamos fazer com que o valor lido pelo sensor de cor seja enviado ao bloco `Display` , que por sua vez exibirá o valor na tela do Brick.

## Instruções de montagem

Para obter as instruções de montagem deste protótipo, no software LEGO® MINDSTORMS® Education EV3, acesse o *Lobby* e clique em `Building Instructions` , `Building Ideas` , `Colour Sensor Down – Driving Base` e, finalmente, em `Open` .

Para a versão anterior do software, estes são os passos: `Robot Educator` , `Building Instructions` , `Colour Sensor Down – Driving Base` e, finalmente, em `Open` .

## Objetivo do programa

O objetivo deste programa é ler a quantidade de luz que está sendo refletida. A princípio, aqui vamos ver apenas como medir a luz refletida pelo sensor de cor, que será a base para um projeto que será feito mais adiante (vamos fazer um Robô Seguidor de Linha), portanto, vamos nos familiarizar, um pouco, com alguns conceitos e lógicas que trabalharemos no futuro.

## O código

Queremos que o programa realize medições a todo instante, por isso vamos utilizar um `Loop` (após o bloco `Start` , claro), assim o programa será executado infinitamente, conforme já visto. O primeiro bloco presente no `Loop` é o `Colour Sensor` , apresentando a seguinte configuração:

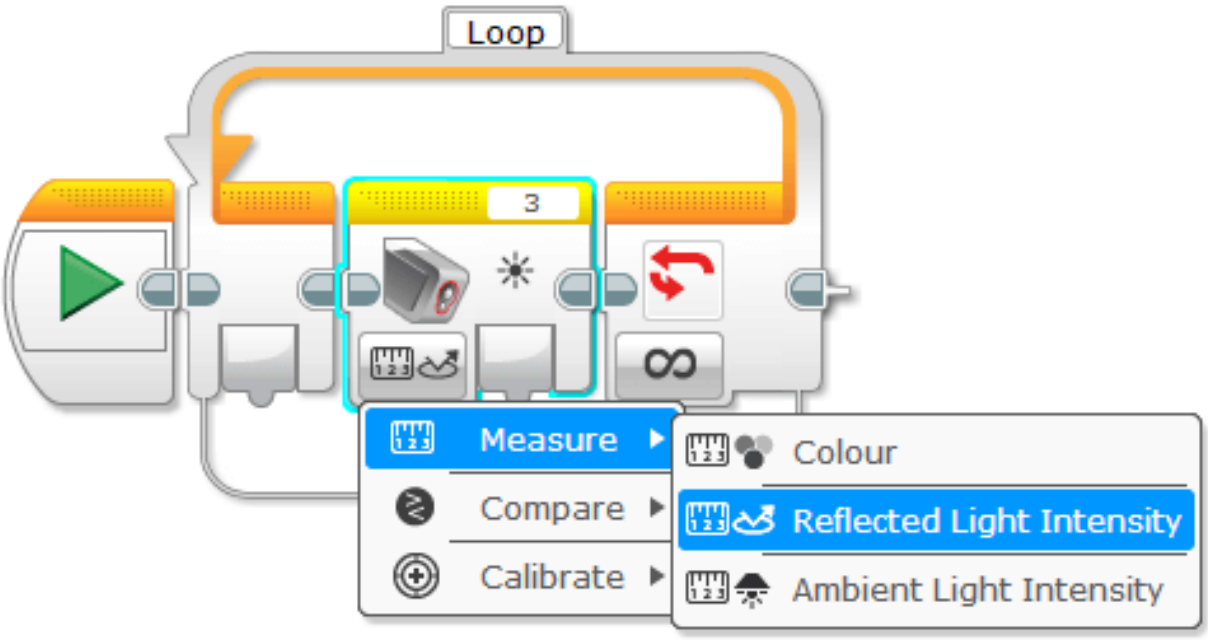



Figura 16.2: Medidor de luz refletida - Passo 1

Ao fazer isso, o valor fornecido pelo bloco na saída `Light` é enviado à entrada `Text` do bloco `Display` , dessa forma conseguimos exibir o valor lido no visor do Brick. É importante lembrar que devemos selecionar a opção `Wired` no bloco `Display` , pois dessa forma pode-se enviar o valor lido pelo sensor diretamente ao bloco `Display` . Vamos utilizar a seguinte configuração:

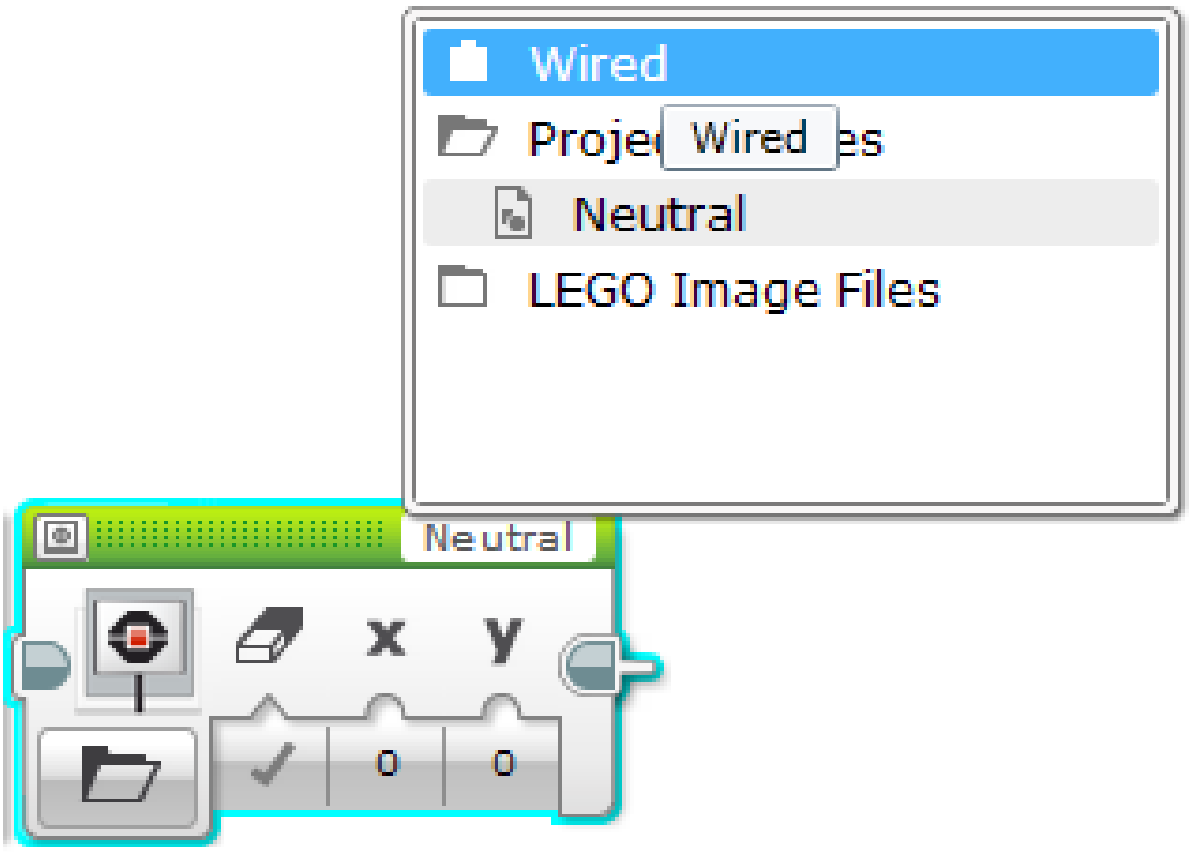



Figura 16.3: Medidor de luz refletida - Configuração do bloco Display

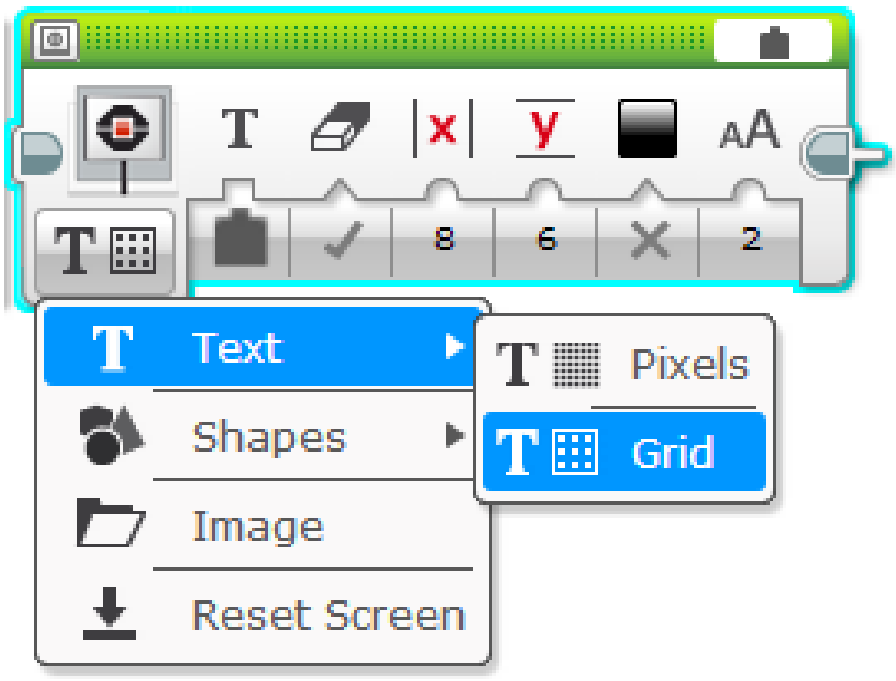



Figura 16.4: Medidor de luz refletida - Configuração do bloco Display

Nosso programa ficou da seguinte forma:

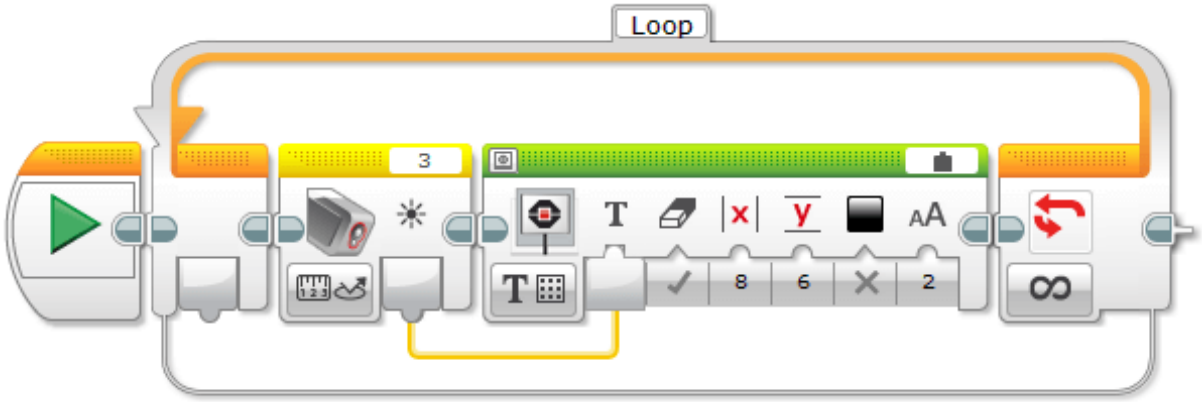



Figura 16.5: Medidor de luz refletida - Código completo

Bem simples, não é mesmo?! No programa a seguir vamos adicionar uma funcionalidade bem bacana que dará ao nosso programa uma interatividade maior.

Este programa apenas exibia o valor lido pelo sensor, e mais nenhuma informação, sendo pouco informativo. Com o uso do bloco `Text` é possível aprimorá-lo, possibilitando a exibição de mais informações na tela.

Esse bloco está localizado na paleta **Data Operations**. Apesar de ter o nome `Text` , esse bloco realiza uma operação. Ele consegue mesclar textos. Na seção a seguir você pode compreender mais detalhadamente a sua função.

## 16.1 O bloco Text

Figura 16.6: Ícone do bloco Text

Apesar de ser um bloco “simples” é muito interessante e útil. Com ele, é possível mesclar textos, podendo combinar até 3 textos. Possui um único modo.

- *Merge* (Mesclar)

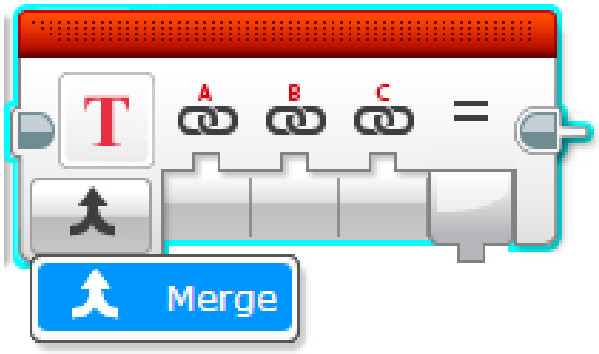



Figura 16.7: Bloco Text - Merge

### Entradas

`A` , `B` e `C` : em cada uma dessas entradas, pode-se inserir um texto. O texto final é exibido na ordem das entradas.

### Saída

`Result` : apresenta o resultado final, ou seja, a união dos três textos.

## 16.2 Utilizando o bloco Text

Da mesma forma que o programa anterior, este programa tem como função informar o valor da intensidade da luz refletida. Entretanto, vamos utilizar o novo bloco que aprendemos: o `Text` . Durante a execução, vamos fazer com que seja exibido na tela `Intensidade (Valor lido pelo sensor)` .

Isso quer dizer que exibiremos na mensagem a palavra "Intensidade", em seguida, o valor fornecido pelo sensor e, por fim, o símbolo .

Na entrada `A` vamos inserir a palavra “Intensidade”. Na entrada `B` vamos conectar o valor lido pelo sensor (por meio do `Data Wires` ). Por fim, na entrada `C` , o símbolo de porcentagem. Dessa forma,

todos os valores inseridos nas entradas serão apresentados como um só texto no `Display` .

Utilizando este bloco, nosso programa ficou da seguinte forma:

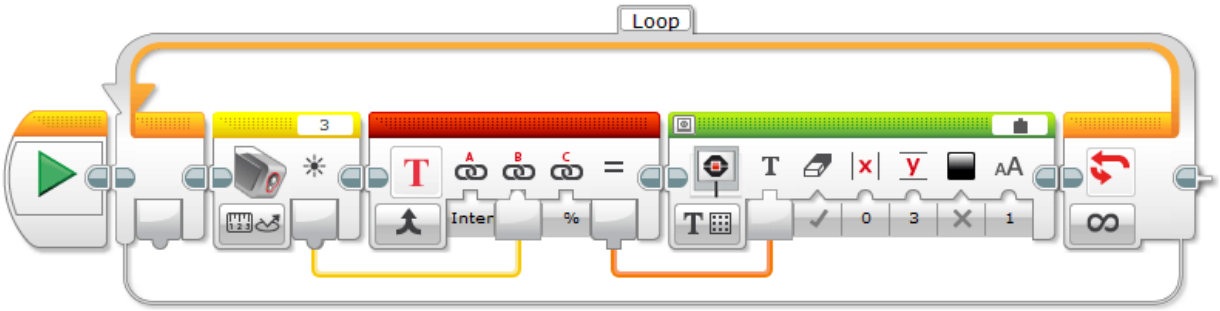



Figura 16.8: Medidor de luz refletida - Utilizando o bloco Text

> **Observação**
>
> Os espaços adicionados ao texto são reconhecidos pelo display, portanto, para que cada texto inserido nas entradas apareça espaçado no display, deve-se adicionar um espaço após o texto.

## 16.3 Tente fazer

Nos programas que fizemos, trabalhamos o modo `Reflected Intensity Light` . Mas, que tal medir a quantidade de luz ambiente? É uma aplicação bastante útil, não é mesmo?! Isso é bem fácil de fazer, você só precisa alterar o modo para `Ambient Intensity Light` . Apenas se lembre de virar o sensor, de modo que ele fique apontado para cima, desta forma você mede a intensidade da luz ambiente com maior precisão.

No próximo capítulo, você já verá como dar vida ao robô e fazer com que ele se movimente. Mas, enquanto isso, você já pode

aproveitar o fato de que tem um robô montado e utilizar os blocos motores para movimentá-lo.

### Dica

Se o seu Brick estiver conectado ao PC, enquanto executa, é possível ver no próprio programa o valor obtido pelo sensor. Isso é feito passando o ponteiro do mouse sobre a saída `Light` do bloco `Colour Sensor` . Em todas as entradas e saídas conectadas por um `Data Wire` você pode utilizar esse recurso. Dessa forma, você pode ver o funcionamento do programa no próprio software.

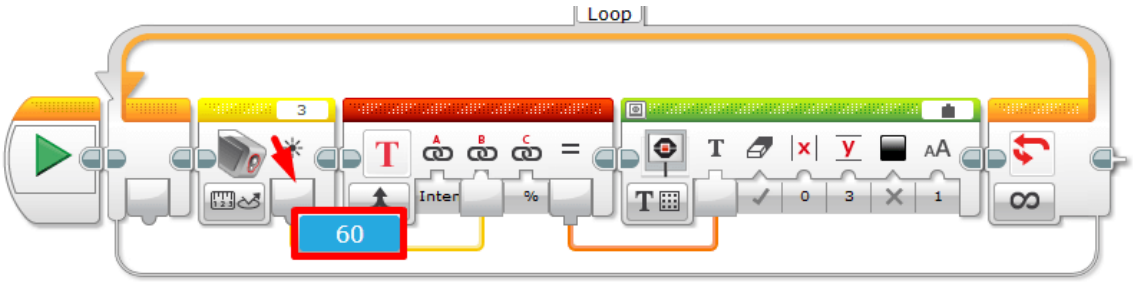



Figura 16.9: Medidor de luz refletida - Saída Light

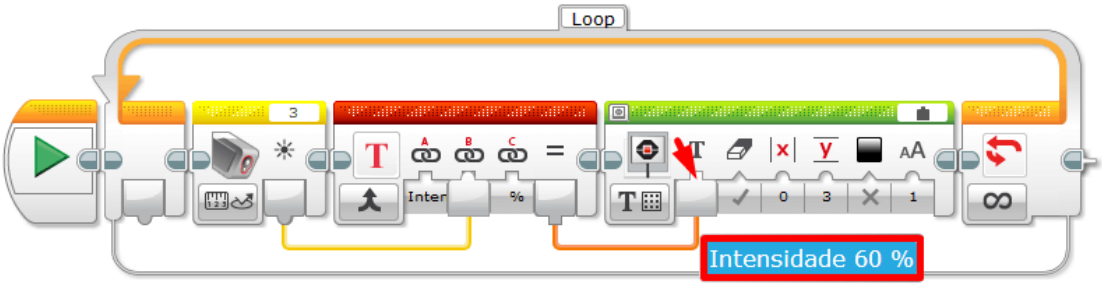



Figura 16.10: Medidor de luz refletida - Entrada B do bloco Text

# Capítulo 17
# Detector de presença

Este programa tem como objetivo detectar presenças ao redor do robô. O robô ficará girando em torno do próprio eixo e, se detectada alguma presença a uma distância menor que 20 centímetros ele emitirá um sinal de alerta.

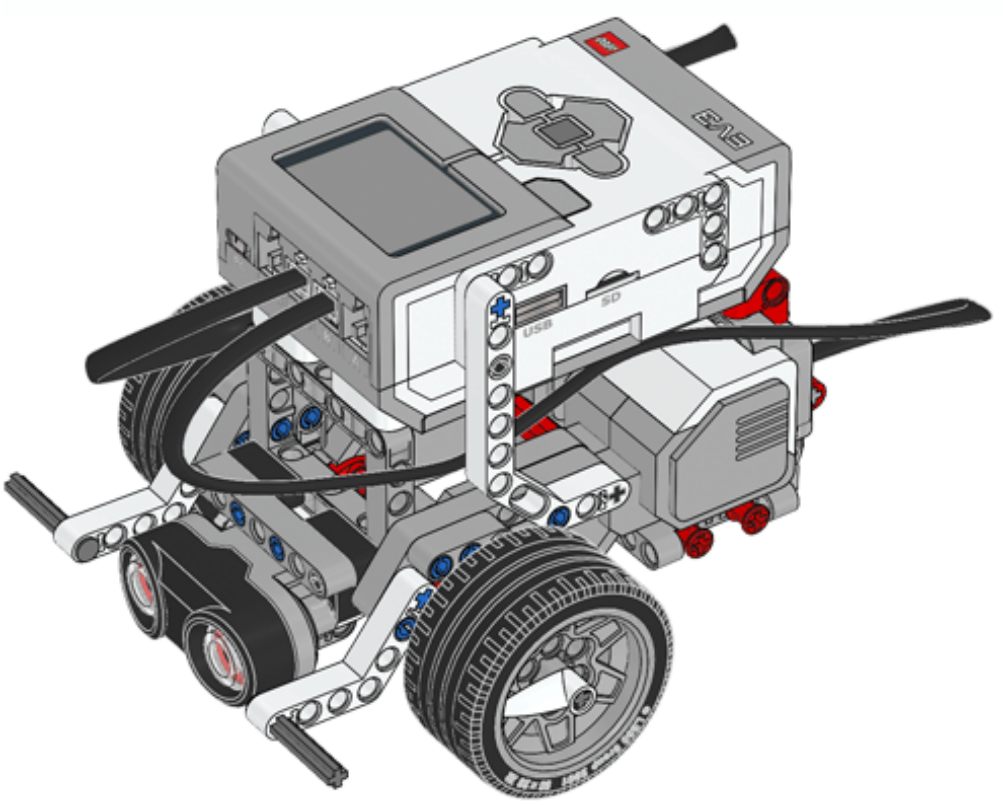


Figura 17.1: Detector de presença

## Instruções de montagem

Caso surja alguma dúvida quanto à montagem, e o usuário esteja trabalhando com a edição Education do EV3, siga os seguintes passos: no *Lobby*, localize a opção `Building Instructions` , em seguida, clique em `Building Ideas` e procure pelo `Ultrasonic Sensor –`

`Driving Base` , quando encontrá-lo, clique nele e, por fim, clique em `Open` para abrir o manual.

Na versão anterior do software, devem-se seguir os seguintes passos: acesse `Robot Educator` , em seguida, `Building Instructions` , clique em `Ultrasonic Sensor – Driving Base` e, finalmente, em `Open` , para ter acesso às instruções de montagem.

## Objetivo do programa

A lógica do programa é a seguinte: o robô ficará girando em sentido horário, caso seja detectado algo a menos de 20 centímetros, ele para (desliga os motores), liga a luz de Status do Brick exibindo a cor laranja com pulsação e reproduz um arquivo de som com a mensagem `Detected` .

## O código

Após o bloco `Start` , devemos colocar o `Loop` , que vai fazer com que o programa seja executado infinitamente. O enunciado dizia que, quando uma detecção for feita, o Brick liga a luz de Status do Brick exibindo a cor laranja. Para que isso aconteça, vamos utilizar o bloco `Brick Status Light` .

## Um pouco mais sobre o bloco Brick Status Light

Figura 17.2: Ícone do bloco Brick Status Light

Com esse bloco controla-se a luz de Status do Brick. Ele possui os seguintes modos de funcionamento:

`Off` - Desliga a luz de Status do Brick.

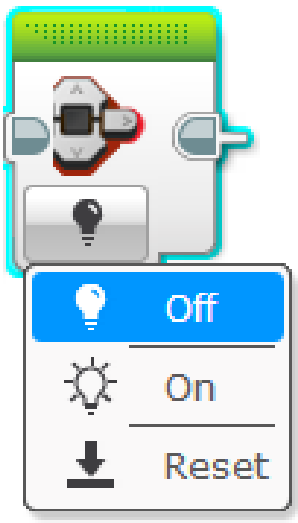



Figura 17.3: Bloco Brick Status Light - Modo Off

`On` - Liga o `Brick Status Light` . Quando ligado, pode-se escolher a cor da luz e a forma como ela será exibida.

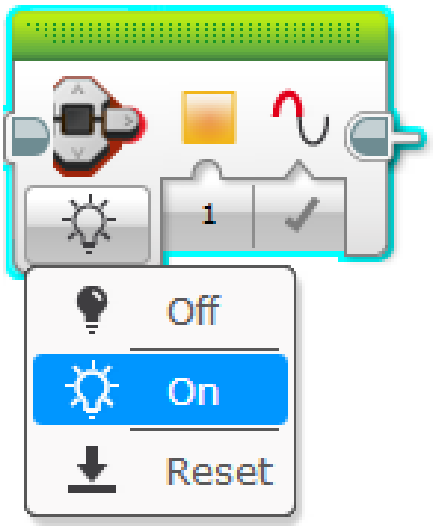



Figura 17.4: Bloco Brick Status Light - Modo On

Apresenta os seguintes parâmetros:

- `Colour` - possibilita a escolha das cores, representadas por quadrados associados à respectiva tonalidade, além de serem identificadas por um número.

Sendo:

0 - *Green* (Verde) 1 - *Orange* (Laranja) 2 - *Red* (Vermelho)

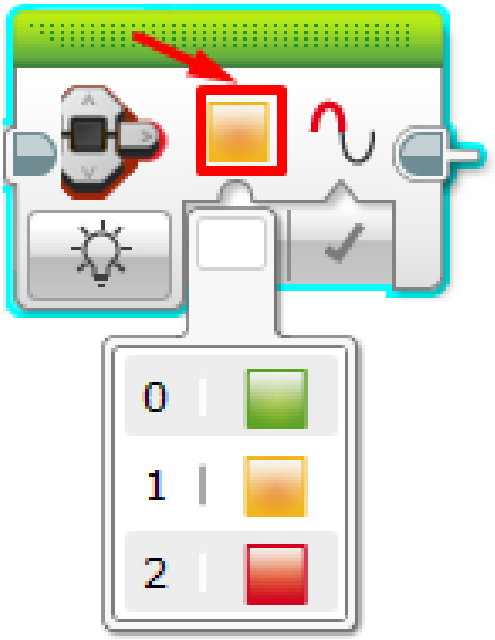



Figura 17.5: Bloco Brick Status Light- Modo On - Parâmetro Colour

- `Pulse` - escolhe-se se haverá ou não a pulsação na exibição da luz.

`True` - A luz é exibida com pulsação. `False` - A luz é exibida de modo contínuo, sem pulsação.

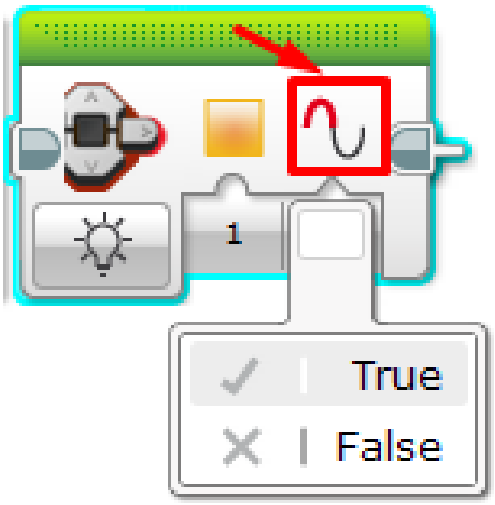



Figura 17.6: Bloco Brick Status Light- Modo On - Parâmetro Pulse

- `Reset` - esse modo ajusta o `Brick Status Light` para suas configurações iniciais.

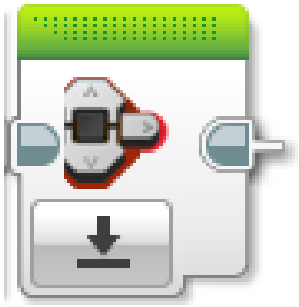


Figura 17.7: Bloco Brick Status Light- Modo Reset

Agora que você já sabe todas as funcionalidades do bloco `Brick Status Light` , vamos voltar ao nosso programa!

Queremos que a luz de Status do Brick seja acionada somente quando a detecção for feita. Por isso, vamos colocar o bloco `Brick Status Light` configurado no modo `Off` , ou seja, estamos desligando a luz de Status do Brick.

Agora vem uma parte simples, porém, muito importante para o robô. Vamos fazer com que o ele se movimente e, conforme o enunciado,

ele deve ficar girando em torno do próprio eixo. Para fazer com que o robô se movimente utilizamos os blocos motores. Então, antes de prosseguirmos com o nosso código, vamos entender um pouco mais sobre esses blocos tão importantes.

## 17.1 Blocos motores

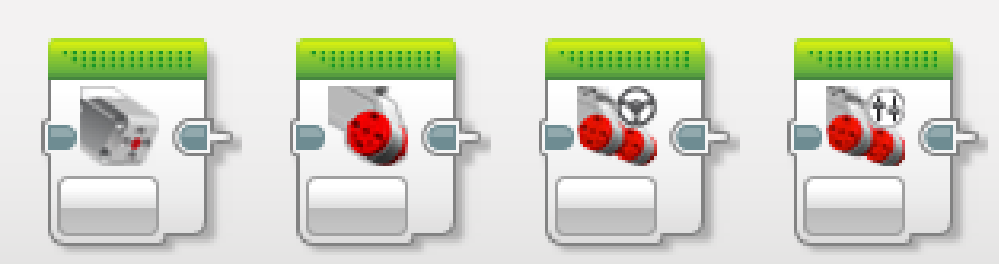


Figura 17.8: ícones dos blocos motores

São os blocos que controlam os motores conectados ao Brick, são eles: `Medium Motor` , `Large Motor` , `Move Steering` e `Move Tank` . Cada um desses blocos possui suas particularidades, mas eles também possuem características semelhantes.

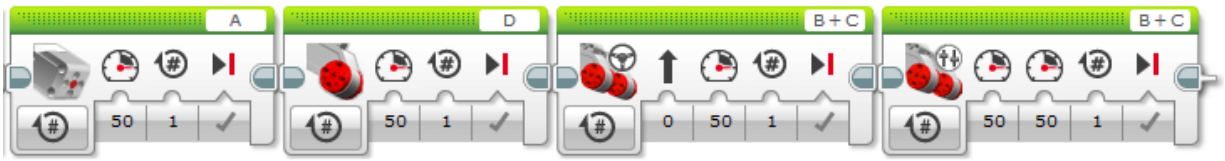



Figura 17.9: Blocos motores, da esquerda para a direita, Medium Motor, Large Motor, Move Steering e Move Tank

Os blocos motores apresentam os seguintes modos de funcionamento:

- `Off`
- `On`
- `On for Seconds`
- `On for Degrees`

- `On for Rotations`

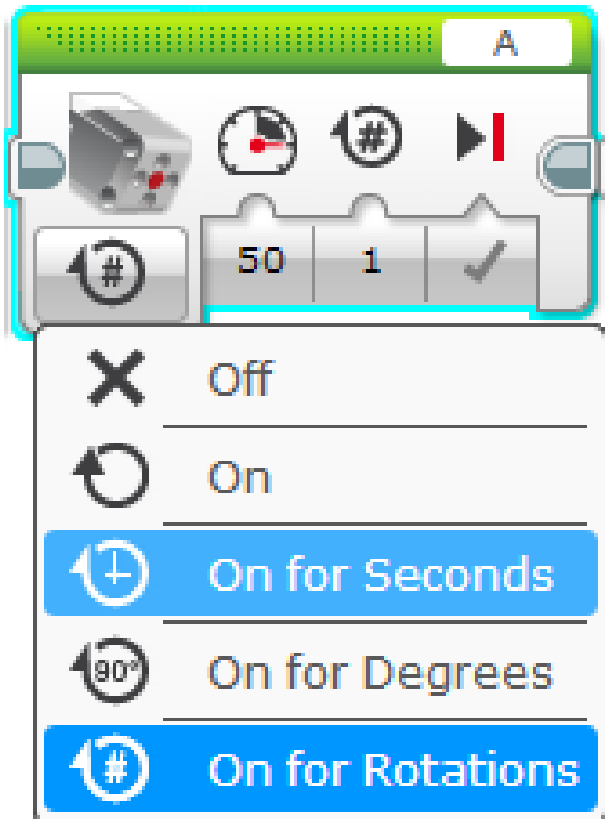



Figura 17.10: Blocos motores - Modos de funcionamento

Esses modos são comuns a todos os blocos motores. É fundamental compreender cada um deles.

### Modo Off

É utilizado para desligar os motores.

O parâmetro `Break at End` pode ser configurado em `True (Brake)` ou `False (Coast)` . No caso `True` , o motor para imediatamente. Caso tenha outro bloco depois dele, o bloco seguinte será executado apenas quando o primeiro terminar completamente seu movimento. No caso `False` , o motor para suavemente, e o bloco seguinte executa sua ação enquanto o motor ainda está parando. O `Brake` simplesmente para o motor, a parada é mais brusca. O `Coast` faz com que a parada de execução do motor seja mais suave.

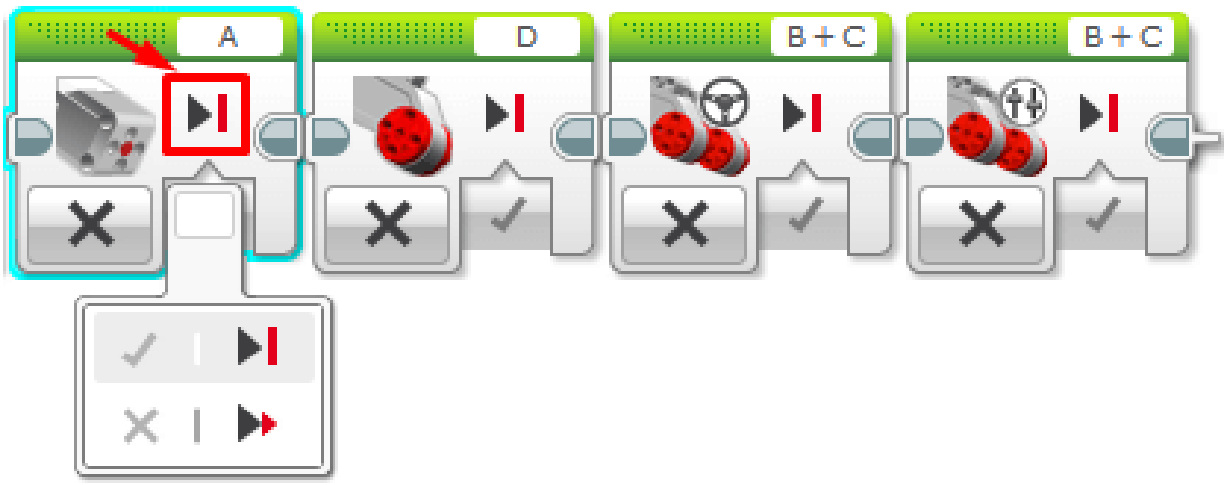



Figura 17.11: Blocos motores - Break at End

## Modo On

O bloco motor é ligado neste modo. Para o `Medium Motor` e para o `Large Motor` configura-se apenas a potência, especificando-a no parâmetro `Power` . O `Move Tank` opera com dois motores, cujas potências são configuradas individualmente. Para o `Move Steering` tem-se apenas um parâmetro de configuração de potência; ele também apresenta o parâmetro `Steering` (Rotação), que é definido em uma escala de -100 a 100. Os números negativos representam o lado esquerdo, e os positivos, o lado direito. Uma volta completa (para direita) corresponde ao valor 100. Caso o movimento fosse para a esquerda, - 100.

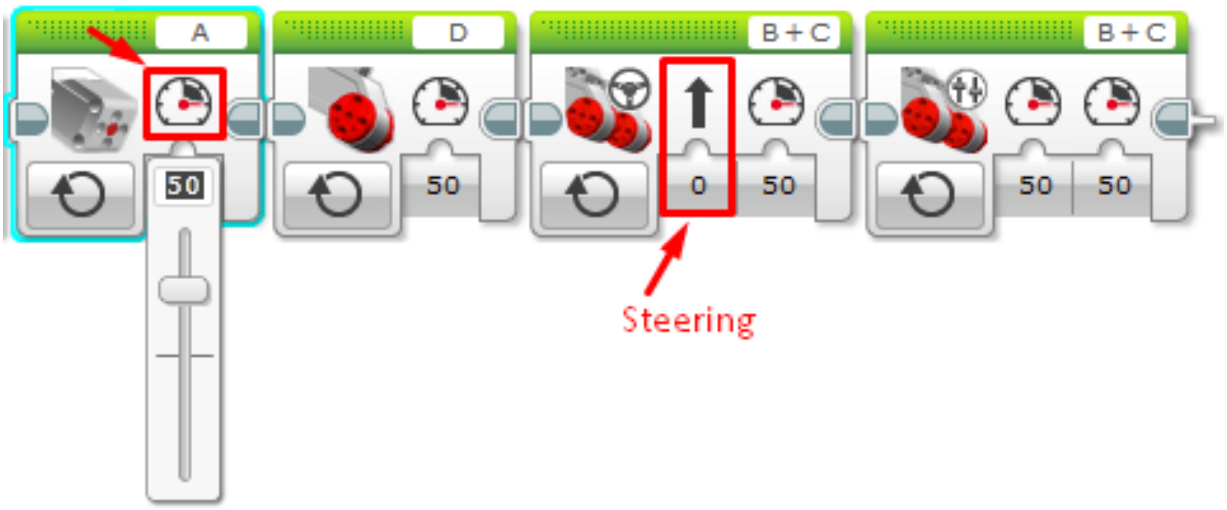



Figura 17.12: Blocos motores - Modo On

## Modo On for Seconds

O motor executa o movimento por um número de segundos definido. O tempo é configurado no parâmetro `Seconds` . Apresenta também os parâmetros do modo `On` (parâmetros de configuração de potência) e do `Off` ( `Brake at End` ).

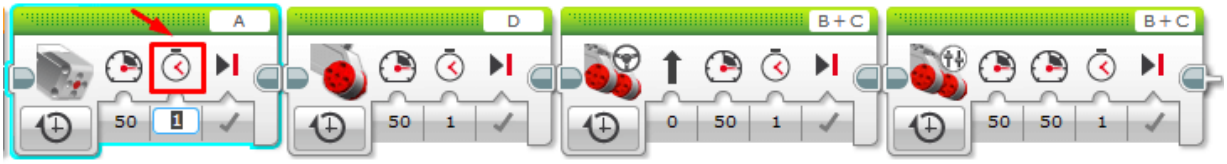



Figura 17.13: Blocos motores - Modo On for Seconds

## Modo On for Degrees

Neste modo, o motor executa seu movimento de acordo com o número de graus, que é definido no parâmetro `Degrees` . Também apresenta os parâmetros `Power` e `Brake at End` .

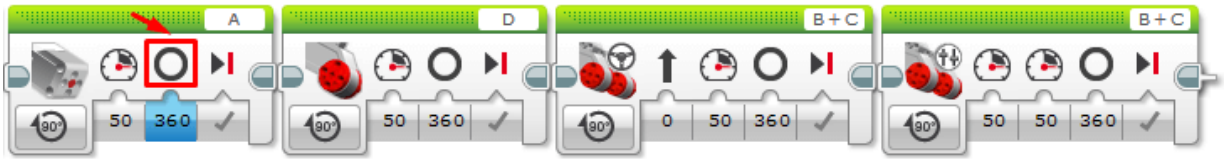



Figura 17.14: Blocos motores - Modo On for Degrees

## Modo On for Rotations

Em vez de graus, a configuração é feita em números de rotações; isso é realizado por meio do parâmetro `Rotations` (360° graus correspondem a uma rotação). Também apresenta os parâmetros `Power` e `Brake at End`.

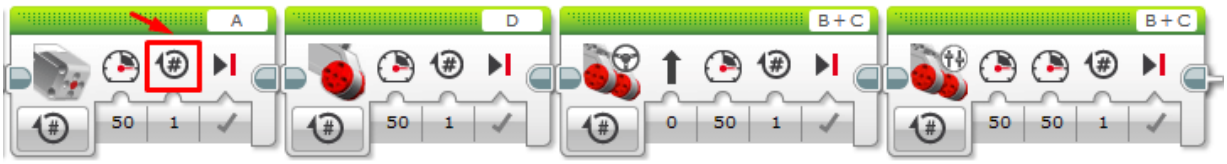



Figura 17.15: Blocos motores - Modo On for Rotations

Agora que você aprendeu sobre todos os tipos de motores, suas funções e modos, vamos dar continuação ao nosso programa.

Para que o nosso robô fique girando em torno do próprio eixo, vamos utilizar blocos motores, mais especificamente o bloco `Large Motor`. Vamos utilizá-los no modo `On`, e conforme acabamos de ver, este modo apresenta apenas o parâmetro `Power`. Vamos configurar os blocos com a mesma potência, com um valor de 25. Para que o robô realize o giro, vamos configurar os blocos motores com potências de mesmo módulo, porém com valores contrários (um positivo e outro negativo). O bloco motor B terá uma potência de -25 e o C de 25. Dessa forma, o nosso robô realizará um movimento rotatório, girando em torno do próprio eixo.

Figura 17.16: Detector de presença - Passo 1

Até aqui o nosso robô só está girando em torno do próprio eixo. Agora vamos adicionar a parte do programa responsável pela detecção de objetos. Vamos fazer isso por meio do bloco `Ultrasonic Sensor` .

## 17.2 O bloco Ultrasonic Sensor

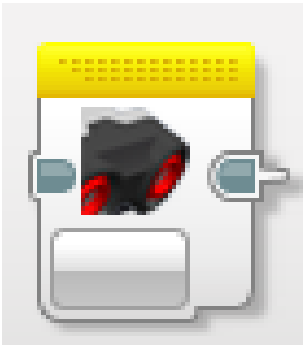


Figura 17.17: ícone do bloco Ultrasonic Sensor

Bloco responsável pelo sensor ultrassônico, pode ser usado no modo `Measure` ou `Compare` . Oferece duas unidades de medida: polegadas ( `Inches` ) ou centímetros ( `Centimetres` ).

## Measure

Utilizado para a medição. A seguir estão as possibilidades de sua utilização:

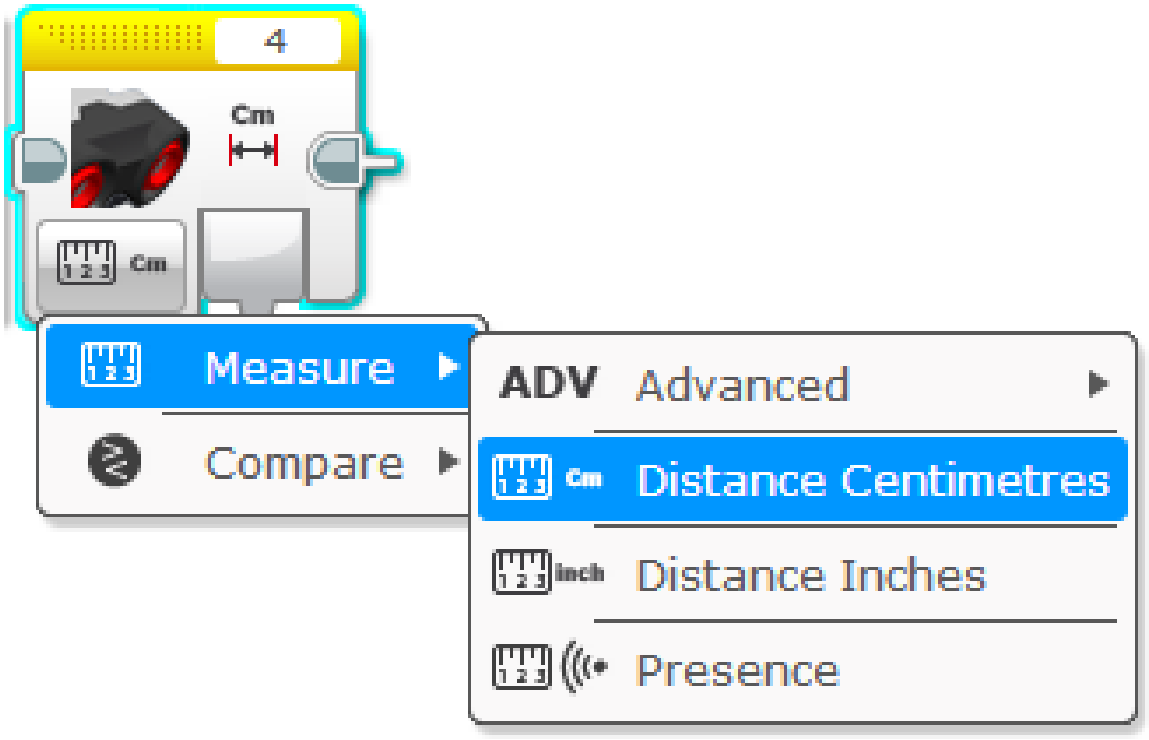



Figura 17.18: Bloco Ultrasonic Sensor - Modo Measure

- `Advanced`

Primeiro, deve-se escolher qual a unidade da medição. Deve-se optar pelo `Distance Inches` ou `Distance Centimetres` , a única diferença entre eles é a unidade de medida.

As medições realizadas nos modos `Distance Centimetres` e `Distance Inches` (descritos melhor a seguir) são feitas com envios de sinais ultrassônicos contínuos. Ao escolher o `Advanced` , é possível selecionar qual o modo desejado. Na entrada `Measuring Mode` é selecionado o tipo de medição, que pode ser:

0 - `Ping` : envio de um único sinal. 1 - `Continous` : envio de sinais contínuos.

A saída pode ser configurada como `Distance in Centimetres` ou `Distance in Inches` , para oferecer o resultado da medição conforme a unidade escolhida.

- `Distance Centimetres`

Este modo realiza a medição em centímetros. Possui apenas uma saída, a `Distance in Centimetres` , que contém o resultado da medição.

- `Distance Inches`

Este modo realiza a medição em polegadas. Possui apenas uma saída, a `Distance in Inches` , que contém o resultado da medição.

- `Presence`

Este modo de medição serve para detectar se há outro sensor ultrassônico no redor. Possui uma única saída. Na saída `Ultrasound Detect` está o resultado da medição, que é um resultado lógico `True` caso haja detecção, e `False` caso contrário.

> **Observação**
>
> É possível conectar dois sensores ultrassônicos e utilizar um deles para medir a presença. Porém, não é possível que os dois realizem a medição de presença um do outro ao mesmo tempo.

## Compare

Utilizado para a comparação. A seguir estão as possibilidades de sua utilização:

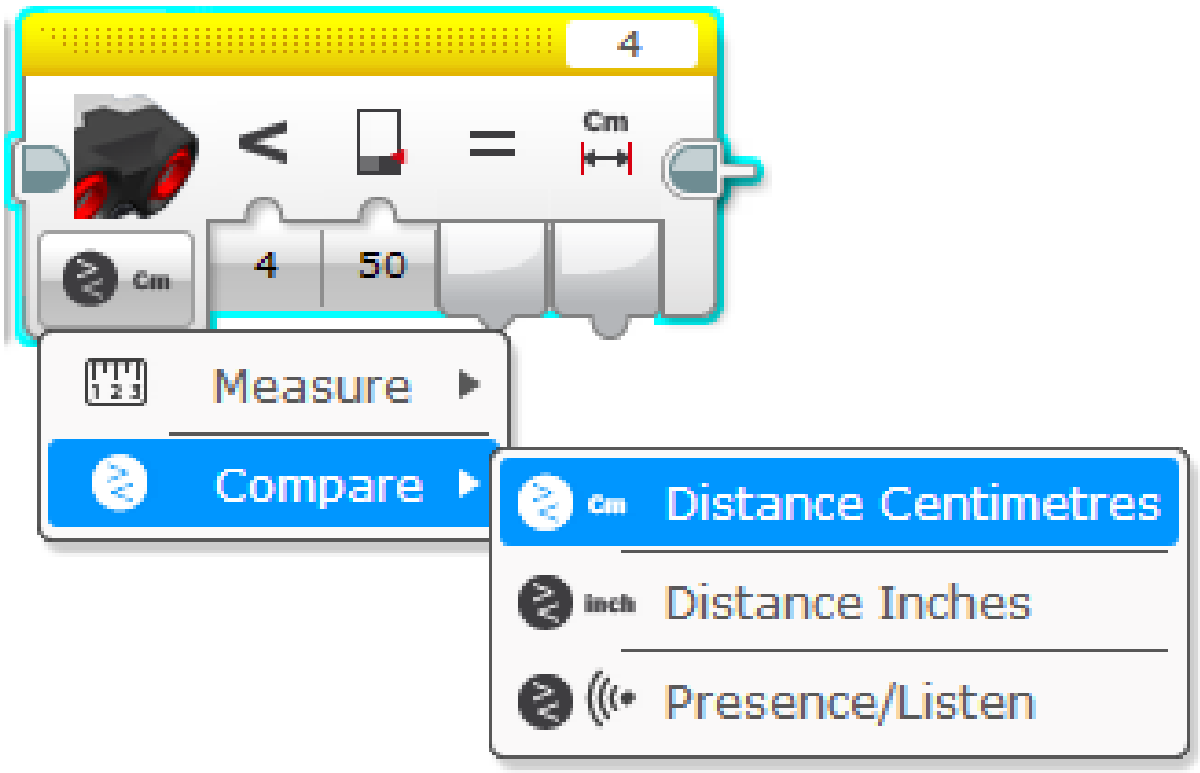



Figura 17.19: Bloco Ultrasonic Sensor - Modo Compare

- `Distance Centimetres`

Este modo tem a função de comparar o valor medido com os parâmetros nele configurados. Nas entradas `Compare Type` e `Threshold Value` são definidos o tipo de comparação e o valor limite, respectivamente. Possui duas saídas, a `Compare Result` e a `Distance in Centimetres` . A saída `Compare Result` oferece o resultado da comparação, um resultado lógico, que pode ser `True` ou `False` . A saída `Distance in Centimetres` contém o valor medido pelo sensor.

- `Distance Inches`

Realiza a mesma função que o `Distance Centimetres` , mas com a unidade em polegadas, conforme o próprio nome já sugere. Portanto, em vez de possuir a saída `Distance in Centimetres` , possui a saída `Distance in Inches` .

- `Presence/Listen`

Neste modo, o sensor detecta presença, mas não envia sinais ultrassônicos, ele apenas “ouve”. Na saída `Ultrasound Detect` , está o resultado da medição, que é um resultado lógico `True` caso haja detecção, e `False` caso contrário.

> **Dica**
>
> Uma forma de saber se o sensor está no modo de presença é observar sua luz. Quando o sensor está no modo presença ele emite uma luz pulsante, enquanto no modo de medição essa luz é contínua.
>
> **Observação**
>
> Não é possível realizar a detecção utilizando dois sensores ao mesmo tempo.

## 17.3 Configurando os blocos sensores

Vamos ver agora dois modos que são comuns e encontrados na maioria deles. São os modos `Measure` (medir) e `Compare` (comparar). Os nomes por si só soam um pouco sugestivos.

Portanto, se você não tem muita experiência em programação, esta seção é imprescindível para a compreensão dos blocos sensores.

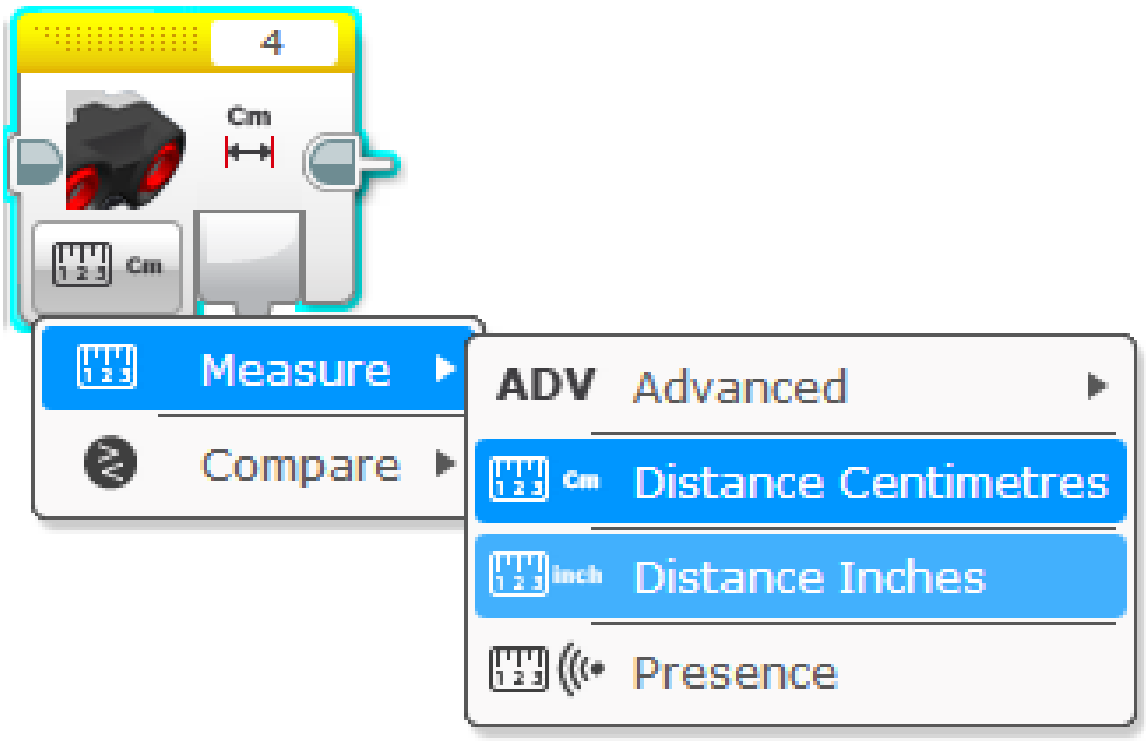



Figura 17.20: Modos Measure e Compare

- `Measure`

Usado para medir, ou seja, o sensor realiza a medição e retorna um valor, que pode ser utilizado pelo programa para um propósito específico.

- `Compare`

Este realiza uma comparação, que é feita de acordo com o parâmetro especificado. Neste modo, além de medir, ele realiza uma operação: a comparação. Na aba `Compare Type` configura-se o tipo de comparação, eles são 6, designados por números de 0 a 5.

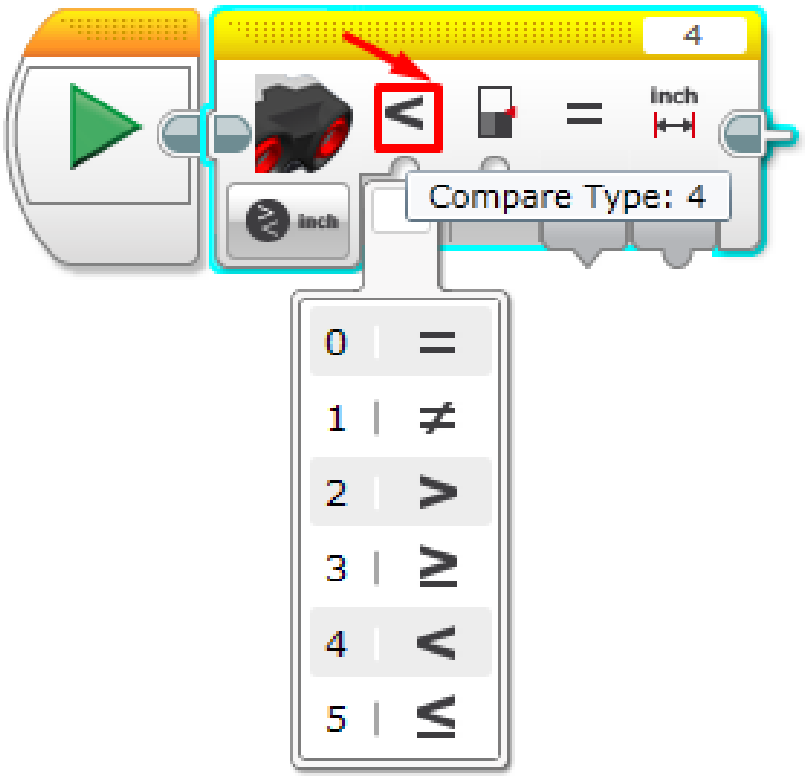



Figura 17.21: Parâmetro Compare Type

- 0 – *Equal To* (Igual a)
- 1 – *Not Equal To* (Diferente de)
- 2 - *Greater Than* (Maior que)
- 3 – *Greater Than or Equal To* (Maior que ou igual a)
- 4 – *Less Than* (Menor que)
- 5 – *Less Than or Equal To* (Menor que ou igual a)

Para que haja a detecção de objetos vamos configurar o bloco `Ultrasonic Sensor` no modo `Compare` . Dessa forma, ele compara a situação atual de acordo com os parâmetros definidos nele.

Vamos utilizar as seguintes configurações: no parâmetro `Compare Type` selecionamos a opção `Less Than` , no parâmetro `Threshold Value` definimos a distância que queremos comparar. Vale lembrar que, o sensor possui um alcance de aproximadamente 250 centímetros.

Portanto, se for detectado algo em uma distância menor que 20 cm, ele gera um valor lógico `True` .

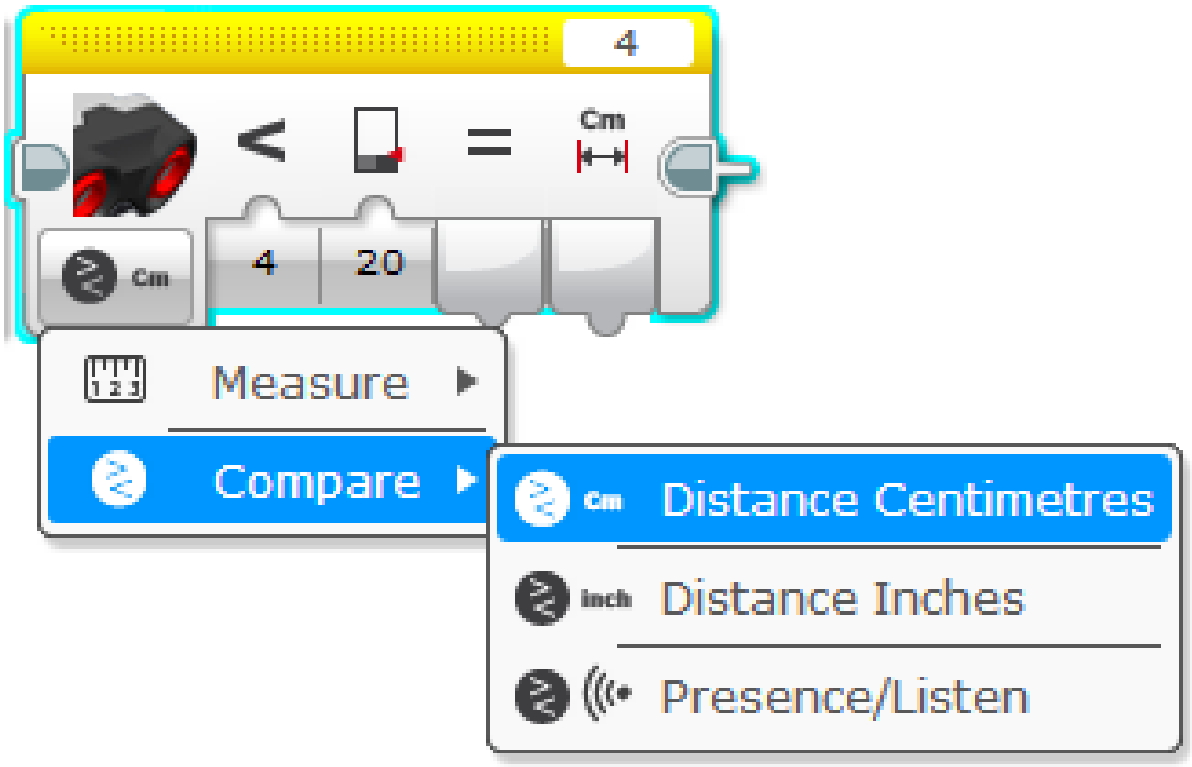



Figura 17.22: Detector de presença - Configuração do bloco Ultrasonic Sensor

O resultado da comparação será enviado ao `Switch` , que devemos configurar no modo `Logic` . Isto é feito ligando o `Compare Result` (saída que contém o resultado lógico da comparação) do bloco sensor ao `Logic` do `Switch` . Esse `Switch` contém as ações que serão executadas pelo robô quando ocorrer a detecção de algo em menos de 20 cm. Os blocos motores são desligados, ou seja, no modo `Off` . Em seguida, o `Brick Status Light` é ativado e, seguindo o enunciado, devemos configurar a cor (no parâmetro `Colour` , e no parâmetro `Pulse` , marcar `True` ). Com isso, quando o robô detectar algo a menos de 20 cm, ele cessará seus movimentos e emitirá uma luz laranja na luz de Status do Brick.

Ainda falta uma coisa. Queremos que o nosso robô emita um som, avisando que o objeto foi detectado. Fazemos isso por meio do bloco `Sound` . Para a nossa felicidade, nosso ambiente de programação nos oferece inúmeros arquivos de som. Estes sons encontram-se na pasta `LEGO Sound Files` , que é acessada por meio do próprio bloco `Sound` . Vamos ver, agora, quais são os passos necessários para realizar essa tarefa.

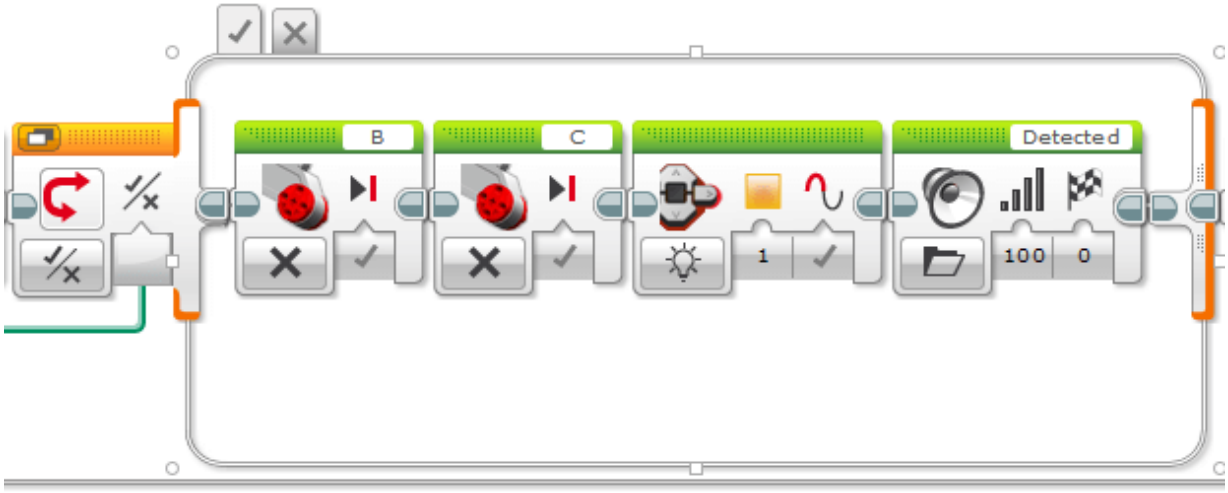



Figura 17.23: Detector de presença - Passo 2

A seleção do arquivo `Detected` é feita da seguinte forma:

No bloco `Sound` selecione a pasta `LEGO Sound Files` .

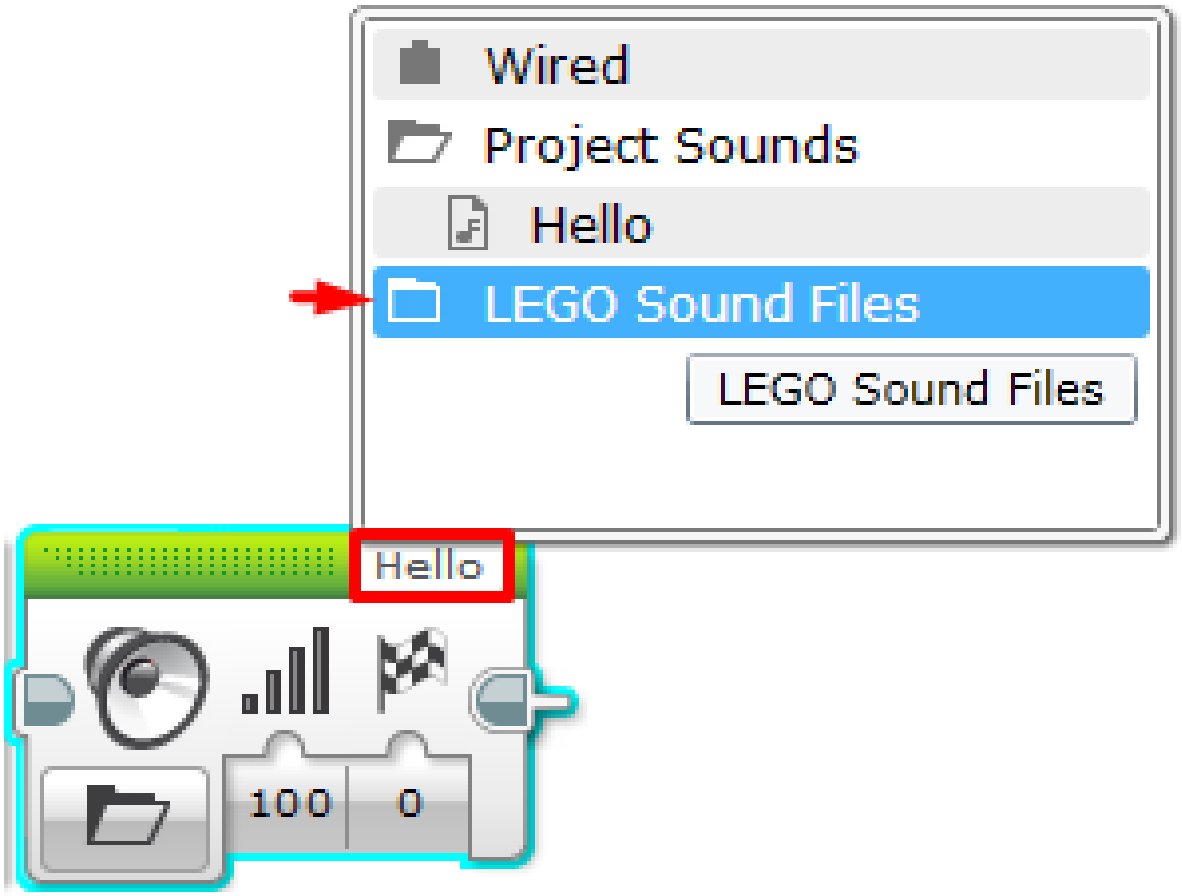



Figura 17.24: Detector de presença - Selecionando o arquivo "Detected"

Em seguida, clique na subpasta `Information` .

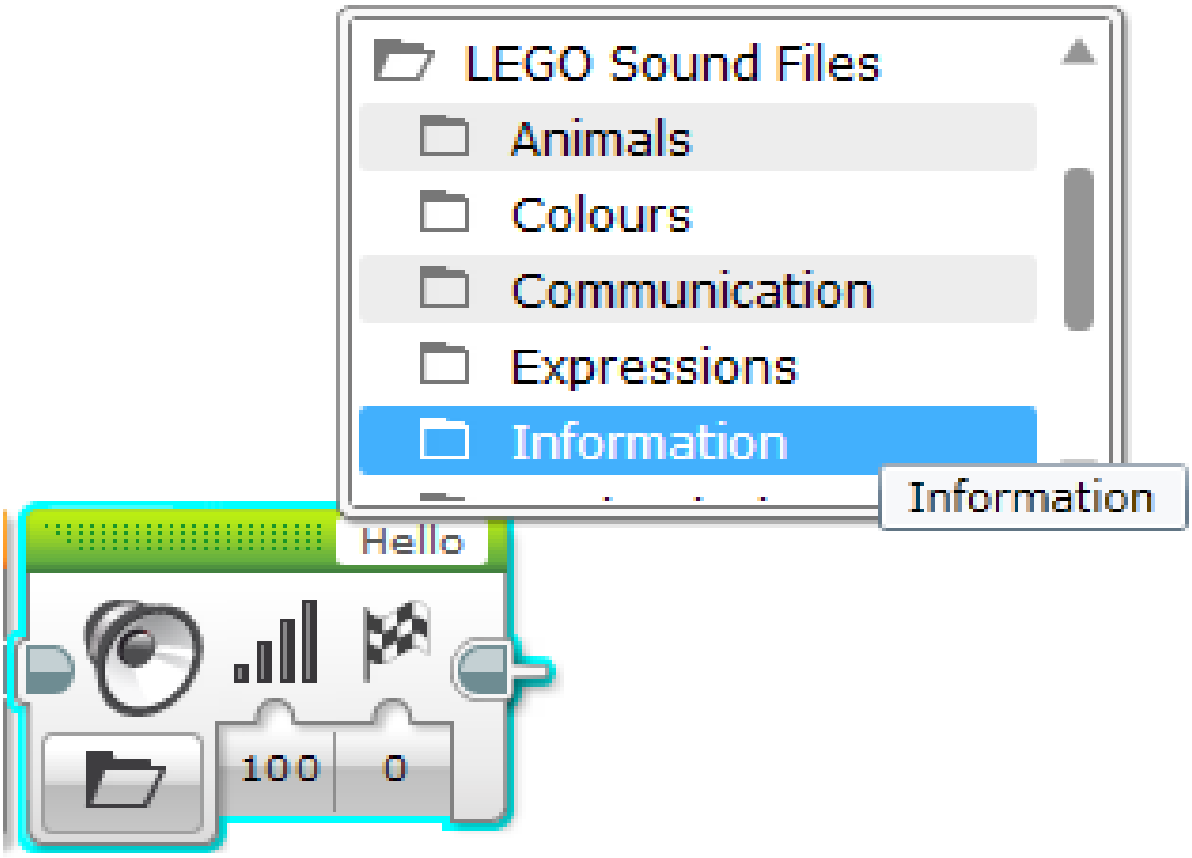



Figura 17.25: Detector de presença - Selecionando o arquivo "Detected"

Por último, clique no arquivo `Detected` .

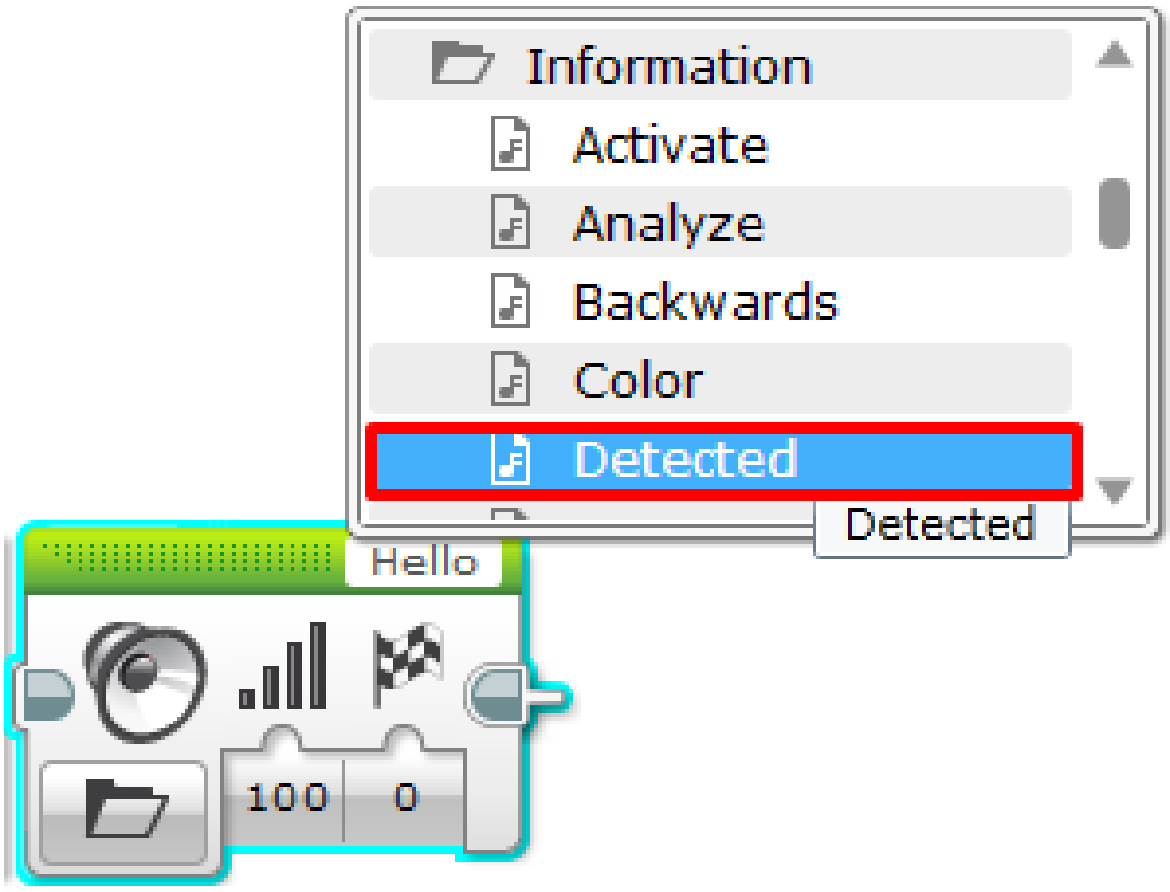



Figura 17.26: Detector de presença - Selecionando o arquivo "Detected"

Esse foi o nosso último passo, o programa está pronto para ser usado!

O programa ficou da seguinte forma:

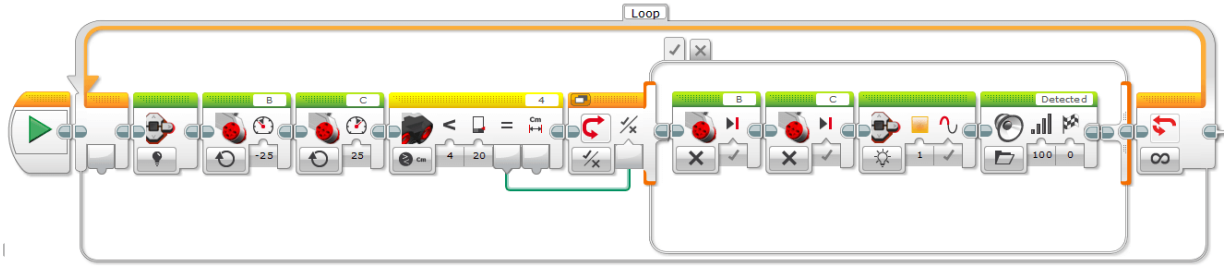



Figura 17.27: Detector de presença - Código completo

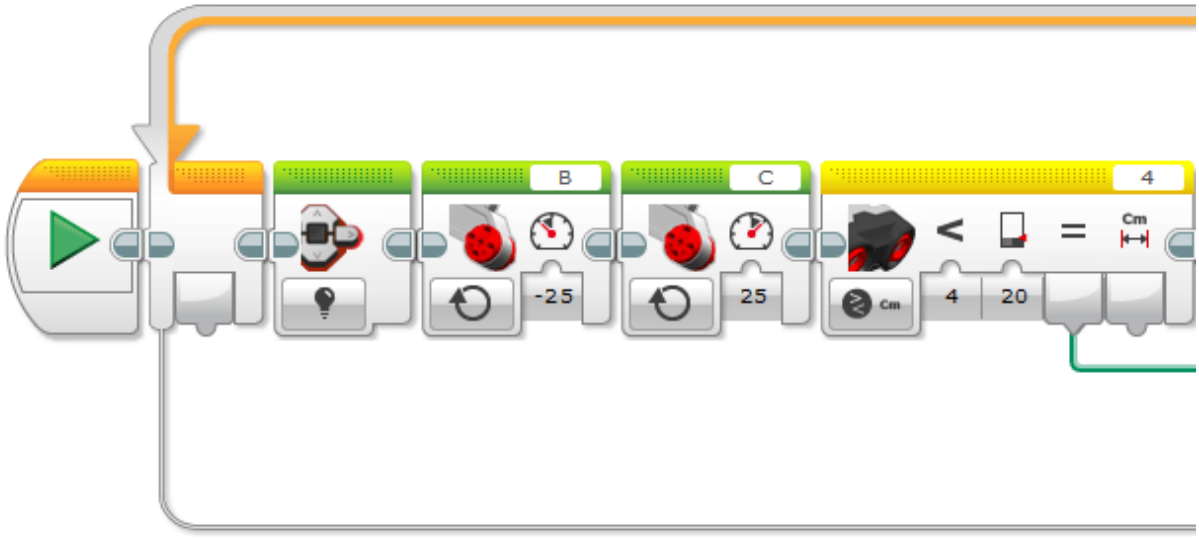



Figura 17.28: Detector de presença - Detalhe 1

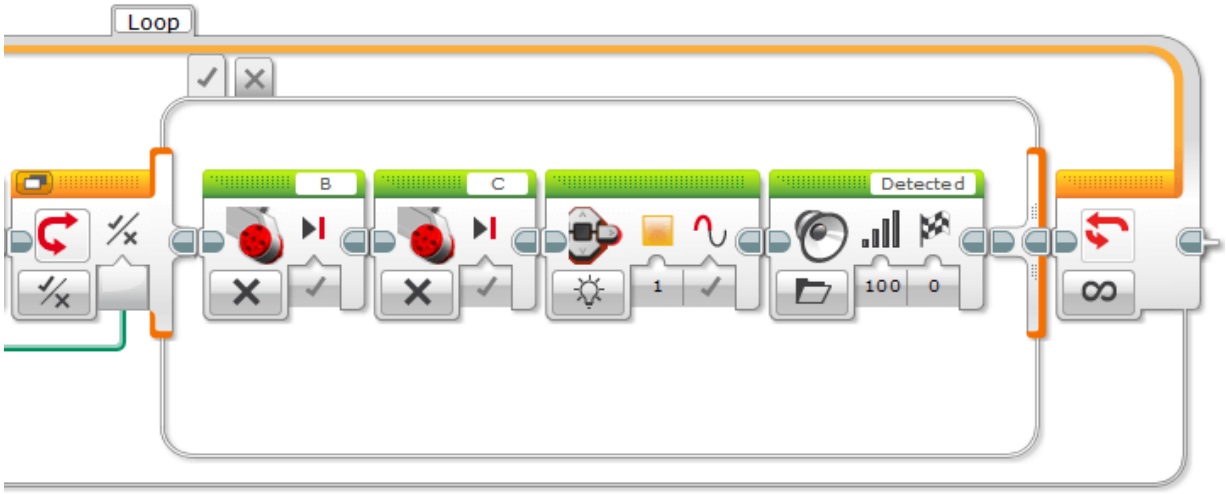



Figura 17.29: Detector de presença - Detalhe 2

## Analisando o programa

Você já sabe que, caso nenhum objeto seja detectado o robô continuará realizando seu movimento giratório. No nosso programa, utilizamos um `Switch` configurado no modo `Logic` . O `Switch - Logic` apresenta 2 `Cases` , um `True` e um `False` . No bloco `Ultrasonic Sensor` é

feita a comparação, e partir dessa comparação obtemos como resultado um valor lógico.

O valor lógico é enviado ao `Switch` e, a partir disso, o `Switch` procede com as ações correspondentes a cada valor. Todas as ações que nosso robô realiza quando um objeto é detectado estão contidas no `Case True` do `Switch` . Mas e o `Case False` ? Nele não colocamos nada, não é mesmo?!

Enquanto um objeto estiver a uma distância maior que 20 cm nosso robô não faz nada, ele simplesmente continua a girar. Por isso, não inserimos nenhum bloco para o `Case False` . Dessa forma, o programa continua executando os blocos anteriores ao `Switch` , pois não há nenhuma instrução (nenhum bloco) contido para o `Case False` .

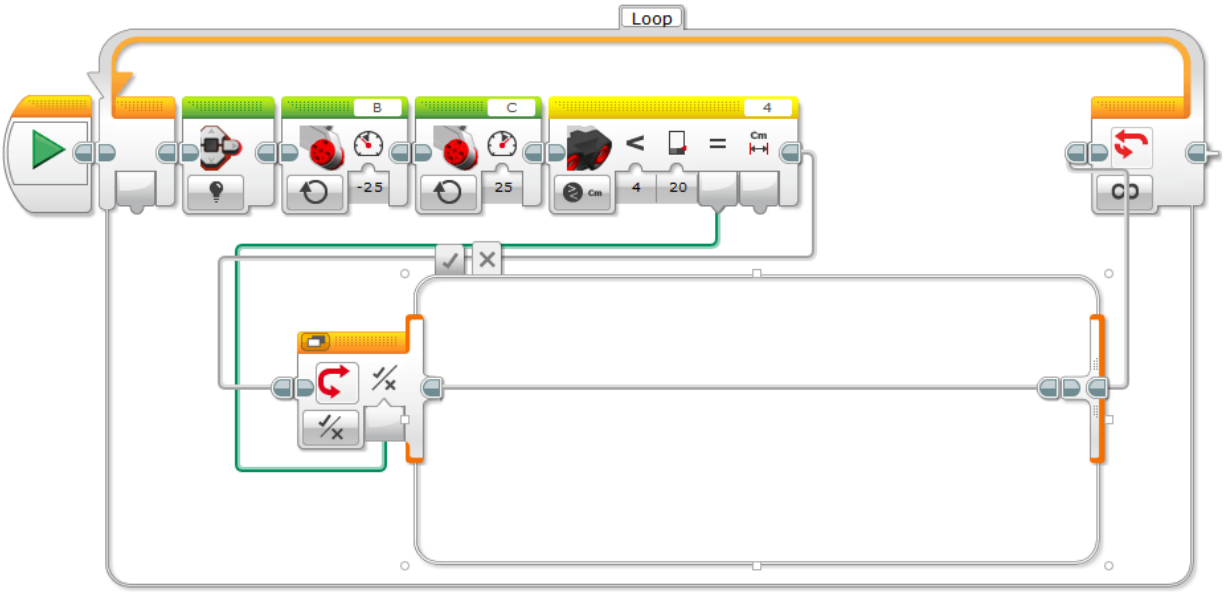



Figura 17.30: Detector de presença - Switch Logic - False

**Observação**

Como você pôde observar, na imagem anterior, o programa estava organizado de forma diferente, seguindo uma organização vertical. É possível programar utilizando não apenas o espaço horizontal, mas o vertical também. Mais adiante, veremos isso mais a fundo.

## 17.4 Utilizando o Switch no modo Ultrasonic Sensor

Agora vamos ver outra maneira de solucionar o mesmo enunciado proposto.

Uma forma mais simples e rápida de programar o robô é utilizando o `Switch – Ultrasonic Sensor` em vez daquela sequência lógica de blocos em que tínhamos o bloco `Ultrasonic Sensor` no modo de comparação ( `Compare` ) que enviava o resultado ao `Switch – Logic` .

Para fazer isso, primeiro, devemos configurar o modo do `Switch` .

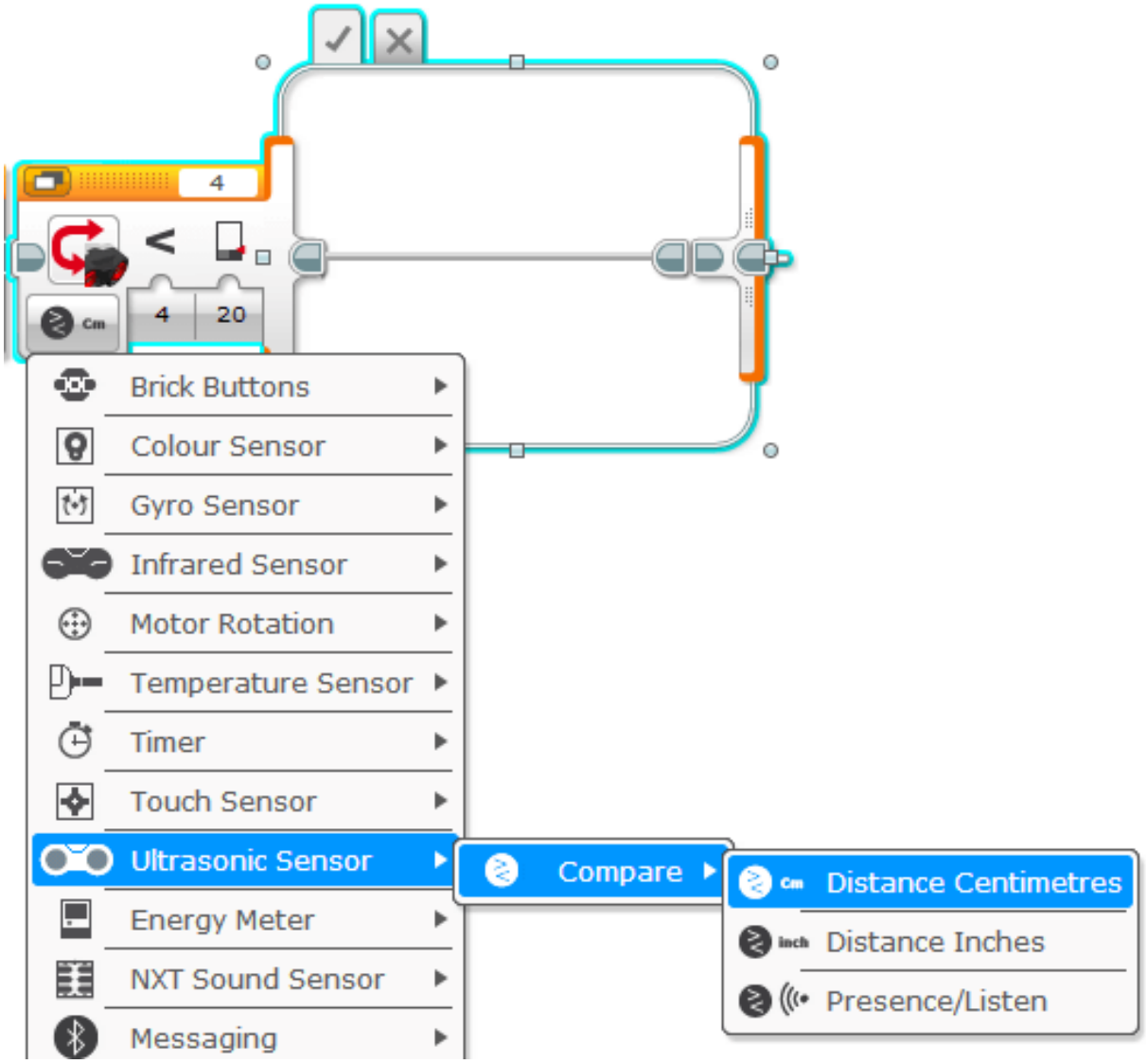



Figura 17.31: Detector de presença com Switch no modo Ultrasonic Sensor - Configurando o Switch

Como no código anterior, esse `Switch` também deve estar dentro de um loop infinito, para que o programa seja executado inúmeras vezes. A linha de cima do `Switch` corresponde ao valor lógico `True` , sendo executada quando o robô detectar algo a uma distância menor que 20 cm.

Na linha inferior (representada pelo "X") temos o `Case False` , que é responsável pelas ações do nosso robô quando não há nenhum objeto em uma distância menor que 20 cm. Aqui vamos inserir os

blocos responsáveis pelo movimento de giro do robô, os mesmos que utilizamos no começo do programa anterior.

O código final fica da seguinte forma:

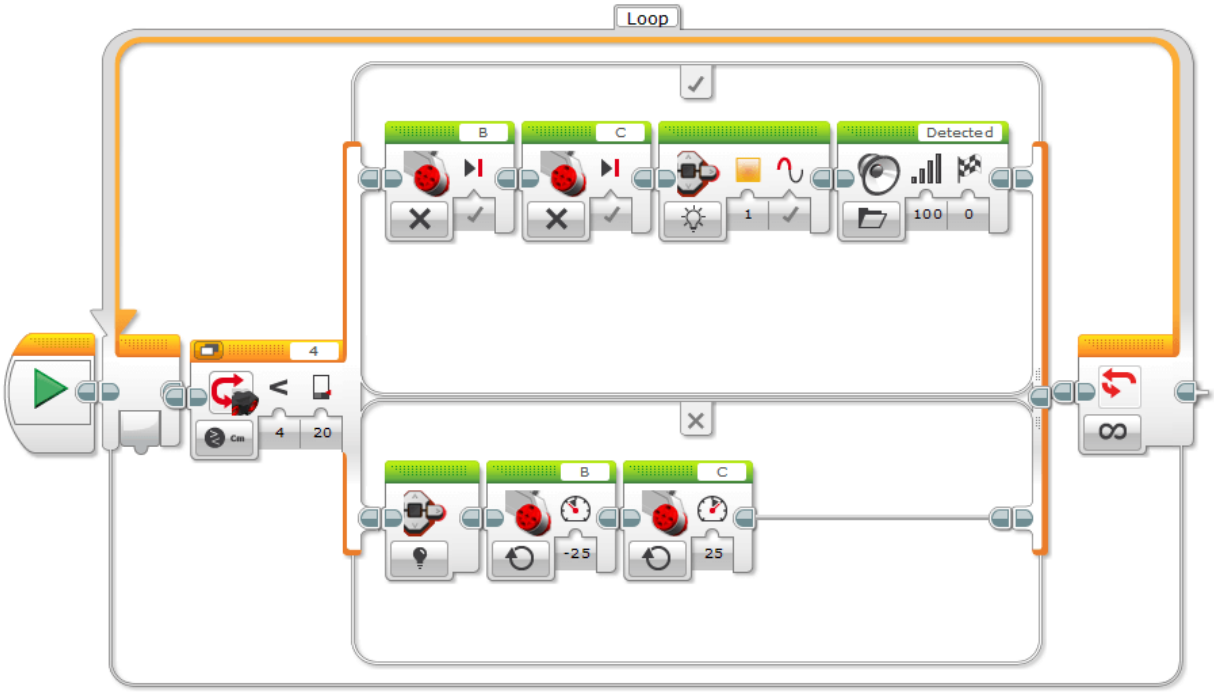



Figura 17.32: Detector de presença com Switch no modo Ultrasonic Sensor - Código completo

Você deve ter notado que o bloco `Sound` apresenta o parâmetro `Play Type` . Utilizamos esse parâmetro para configurar a forma como o bloco deve proceder com a execução.

## Tipos de execução do bloco Sound

O bloco `Sound` apresenta os seguintes tipos de execução:

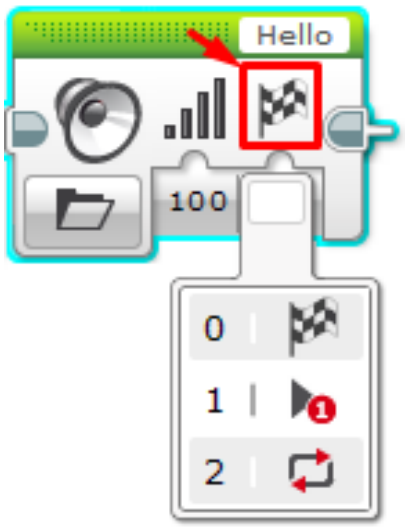



Figura 17.33: Bloco Sound - Parâmetro Play Type

- 0 – *Wait for Completion*: a próxima ação designada após o bloco `Sound` só é executada quando este termina a sua ação.
- 1 – *Play Once*: toca o arquivo apenas uma vez.
- 2 – *Repeat*: o bloco repete indefinidamente a execução do arquivo.

> **Dica**
>
> O software possui um AUTO-ID. Isso significa que ele tem definições automáticas das portas, as quais podem ser modificadas. O bloco `Medium Motor` está configurado na porta A. O `Large Motor` na porta B. Os blocos `Move Tank` e `Move Steering` nas portas B e C. Isso pode ser interessante para a montagem do robô, pois utilizando a configuração padrão não é necessário se preocupar com a mudança das portas durante a programação, mas isso é apenas um detalhe.

## 17.5 Tente fazer

Concluímos o nosso último programa. Neste capítulo, aprendemos sobre o bloco `Brick Status Light` , vimos todos os blocos motores, com os modos de operação de cada um. Agora, que tal testar suas habilidades e colocar em prática todo esse conhecimento?

No nosso programa, ligamos o `Brick Status Light` somente quando o objeto fosse detectado, e quando nada era detectado, desligávamos o bloco. Adicione a luz de Status para o Brick exibir mesmo caso nenhum objeto seja detectado. Escolha a cor que preferir, mude também o tipo de exibição do Status para uma exibição contínua, sem pulsação.

Você também viu que existe um bloco específico que permite ao robô realizar giros: o `Move Steering` . Em vez de usar dois blocos `Large Motor` , utilize um bloco `Move Steering` para realizar o giro de seu robô.

Nos capítulos anteriores, aprendemos como exibir mensagens na tela do Brick. Implemente no seu programa a seguinte função:

- Quando um objeto for detectado, deverá ser exibida a mensagem "Detected" na tela do Brick.

# Capítulo 18
# Conclusão

Mais uma etapa concluída, parabéns!

Nesta parte do livro foram apresentados nossos primeiros programas. De forma prática, você aprendeu a função dos principais, e também mais utilizados, blocos de programação.

Por meio de programas bem simples, trabalhamos vários conceitos e blocos. Você aprendeu a utilizar os blocos de controle de fluxo. Um dos blocos que mais utilizamos foi o bloco `Loop` , que permite encapsular toda uma sequência de blocos e repeti-la até mesmo infinitas vezes. Aprendeu também a fornecer ao programa instruções mais sofisticadas, como esperar por algo ou tempo ( `Wait` ). Vimos, também, que é possível fornecer escolhas ao programa utilizando o bloco `Switch` , viu que ele pode ser utilizado em diversos modos.

Também foram estudadas as variáveis. Vimos na prática o quão importante e útil é uma variável. Para tal, utilizamos o bloco `Variable` , que pode ser configurado para armazenar, ou informar o valor armazenado. Outro aspecto muito importante que trabalhamos foi a utilização dos blocos sensores, como o bloco `Ultrasonic Sensor` e o bloco `Colour Sensor` . Você aprendeu como comunicar os dados fornecidos por esses sensores às variáveis e às estruturas de controle. Ou seja, fomos interligando todos os conhecimentos adquiridos para deixar o programa com mais funções ainda.

Esta parte do livro lhe forneceu uma base, um pontapé inicial na programação. Nos próximos capítulos utilizaremos esses conhecimentos já estudados como base para a criação de novos programas, ainda mais complexos e desafiantes.

# EXPERIÊNCIAS

Nos capítulos anteriores, fizemos os nossos primeiros programas, que servirão de base para esta próxima etapa. Nesta parte serão feitos programas mais complexos e desafiadores, com mais aplicações práticas (e mais divertidos também). Desta vez, além de trabalhar os programas e suas variações, também serão propostas atividades. Assim você pode treinar todos os conceitos estudados e aperfeiçoar suas habilidades como programador. Contudo, caso tenha alguma dúvida ou algo não saia como conforme o esperado, não se preocupe. Ao final do capítulo você encontrará as respostas para as atividades, não apenas o programa completo, mas sim, a resolução comentada e explicada de cada trecho.

Faremos as seguintes experiências:

- **Robô Seguidor de Linha**
- **Robô Sumô**
- **Robô Explorer**

Os programas desenvolvidos nesta parte podem ser acessados na seguinte pasta:

https://github.com/vitor000x/ev3_Codes/tree/master/Experiencias.

Essa pasta contém pastas relativas a cada uma das três experiências que faremos ao longo desta parte do livro.

Alguns programas são bem grandes, sendo assim, ocupam bastam espaço horizontalmente, o que torna inviável adicionar uma imagem do programa completo. Para que o programa inteiro caiba na imagem é preciso aplicar um "Zoom Out" muito grande, que acaba por minimizar demais os blocos.

Logo, durante a explicação de alguns códigos e/ou, durante a resolução das atividades, é aconselhável que você acompanhe a explicação do programa acessando a pasta referente ao que está sendo trabalhado. Desse modo, você poderá sanar de forma

imediata alguma dúvida quanto à configuração de um bloco ou a construção de um programa.

Prepare-se, vamos começar!

> **Atenção**
>
> Fique sempre atento à identificação da porta no qual você conectou um motor ou um sensor. Na hora de fazer o download do programa para o Brick, sempre se certifique de que os blocos utilizados para controlar os motores e sensores estão com a identificação correta.

# Capítulo 19
# Seguidor de Linha

Você se lembra do "Medidor de luz refletida"? Naquele capítulo que abordou o programa, foi dito que mais adiante seria construindo um Robô Seguidor de Linha. Pois é, a hora chegou. Por isso, você pode utilizar aquelas instruções para montar o seu robô, ele servirá bem para os nossos primeiros programas.

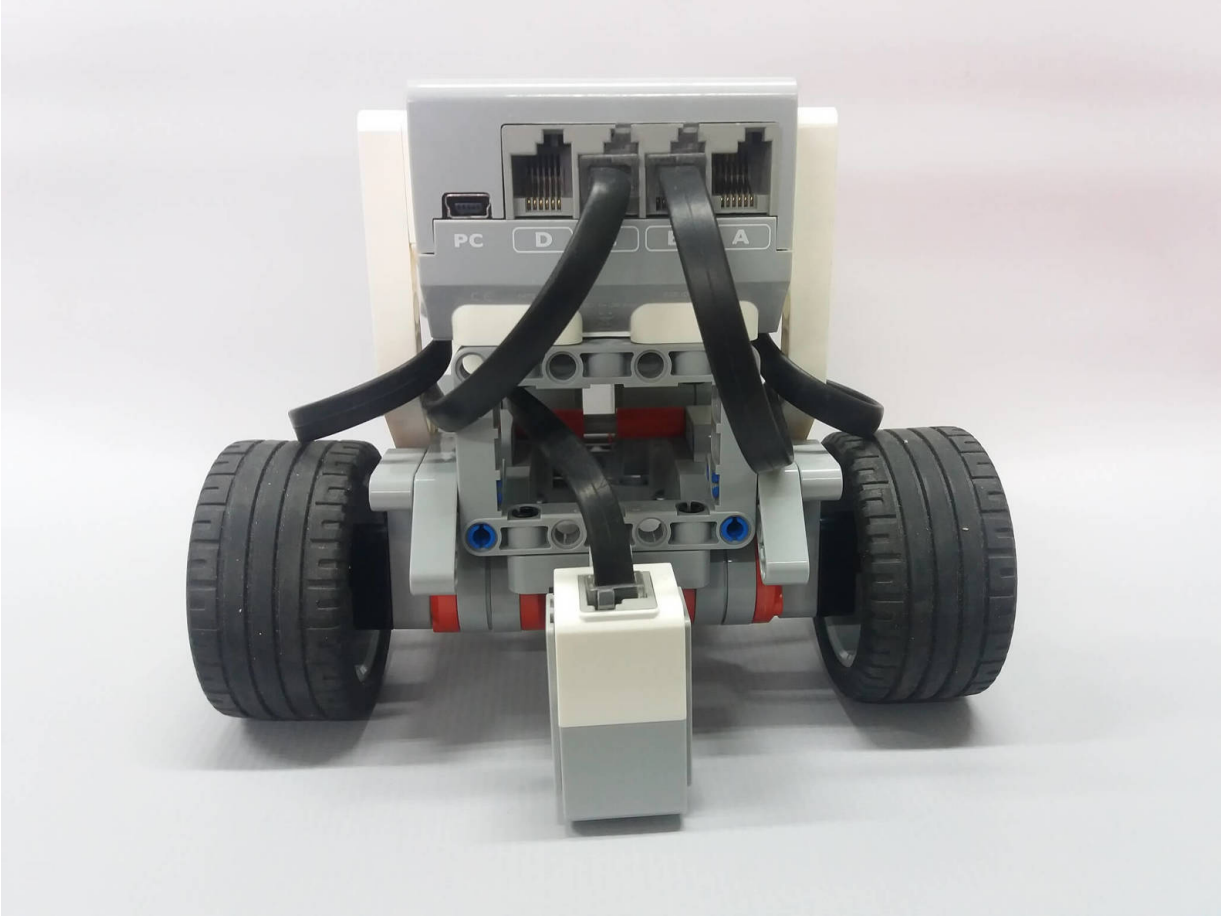



Figura 19.1: Seguidor de Linha

## Instruções de montagem

Esse robô foi montado no capítulo *16. Medidor de lu refletida*, então, para montá-lo, basta consultar o capítulo e seguir as instruções.

Mas para resolver as atividades a seguir, vamos precisar realizar algumas alterações no nosso robô. É fundamental mudarmos a posição do sensor de cor, colocando-o aproximadamente no meio do robô. Outra alteração com que é preciso ter cuidado é em relação à altura do sensor de cor. Se ele estiver muito acima da superfície, a leitura do valor refletido pode ser comprometida. Por isso, é importante que você mantenha seu sensor de cor próximo à superfície de leitura.

As instruções de montagem para realizar a modificação estão disponíveis na pasta `Instructions` desta experiência, cujo link de acesso direto é:

https://github.com/vitor000x/ev3_Codes/tree/master/Experiencias/1-Seguidor-de-Linha/Instructions.

**Objetivo do programa**

O objetivo da nossa primeira experiência é desenvolver um programa que movimente o robô ao longo de uma linha negra desenhada sobre um fundo branco, conforme ilustra a figura a seguir.

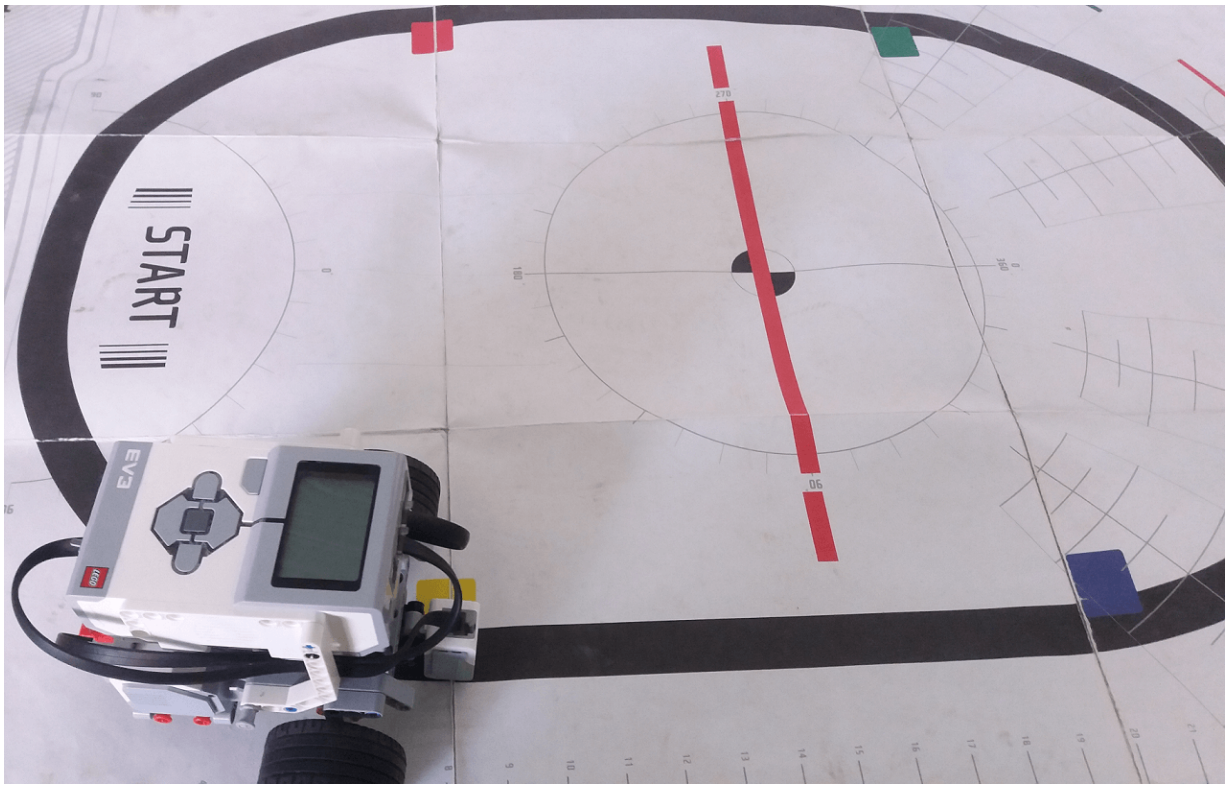



Figura 19.2: Pista (que acompanhava o kit anterior, o NXT) com o Robô Seguidor de Linha

Mas, caso você não tenha uma pista, isso pode ser resolvido facilmente. Você pode criar a sua própria pista, basta usar uma fita adesiva (preferencialmente preta) e fixar em uma superfície. Para a

construção da sua pista, use a foto anterior como referência, também, fique atento em relação às curvas.

Como você percebeu, na imagem da pista, há um perímetro escuro traçado na pista. Nosso objetivo é programar o robô para que siga esse traçado. Dessa forma, o traçado será a trajetória.

A ideia do programa seguidor de linha é criar dois blocos motores, associando um deles à roda esquerda e o outro à roda direita. Para o movimento de zigue-zague vamos utilizar potência baixa num dos motores (10% do máximo) e potência média no outro (50% do valor máximo).

**Medindo e analisando a quantidade de luz refletida**

Com o seu robô ligado, é possível fazer uma análise prévia da quantidade de luminosidade refletida na parte clara e no traçado escuro da pista. Assim, você pode configurar seu robô da melhor forma possível. Vamos ver como fazer isso:

O primeiro passo é ligar o Brick, abrir seu programa e localizar a *Hardware Page* (situada no canto inferior direito). Feito isso, clique na opção `Port View` , pois assim poderemos visualizar os valores medidos pelo sensor em tempo real.

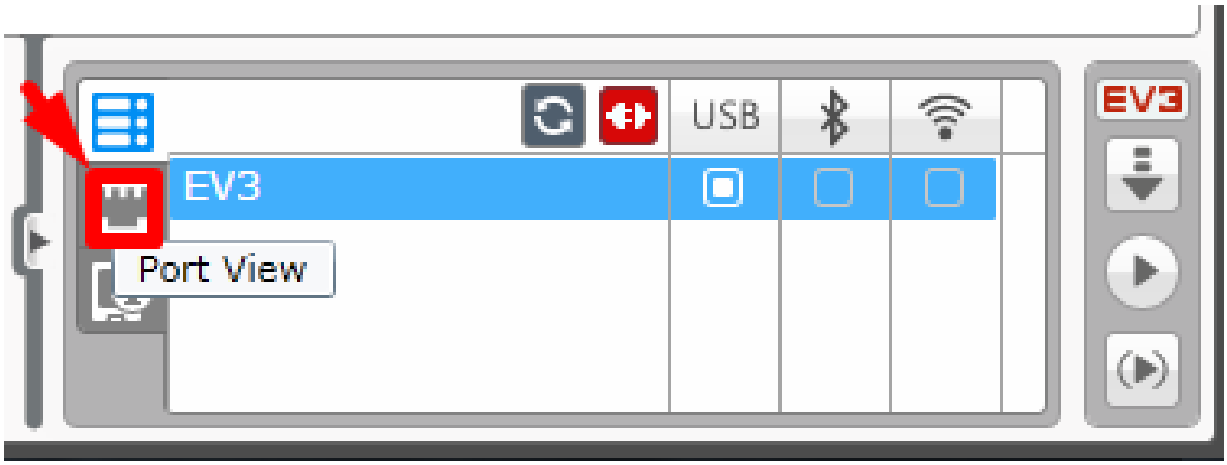



Figura 19.3: Hardware Page - Port View

Primeiro, vamos posicionar o robô para medir o valor refletido na linha negra, conforme vimos na primeira imagem. A seguir vamos conferir o valor medido.

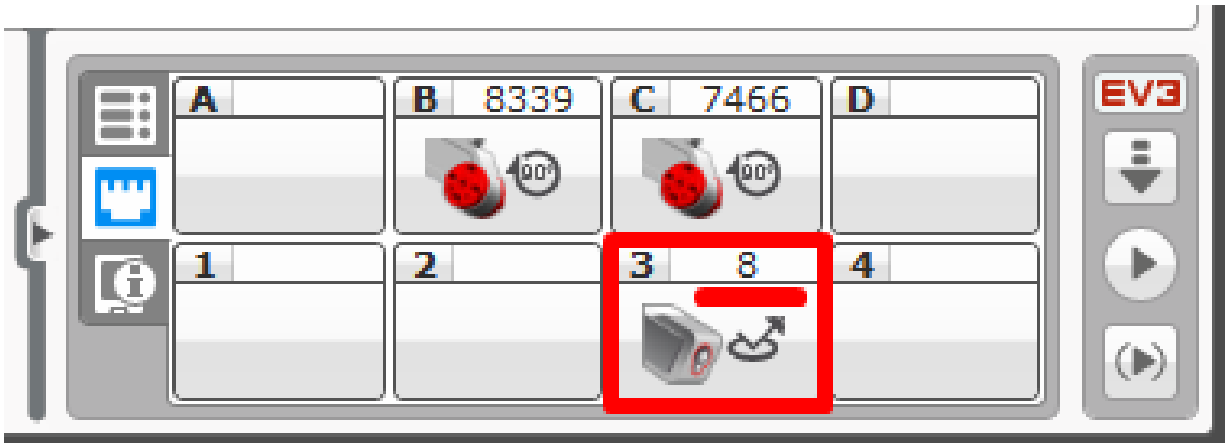



Figura 19.4: Valor medido na linha negra

Vamos conferir o valor medido na parte branca da pista:

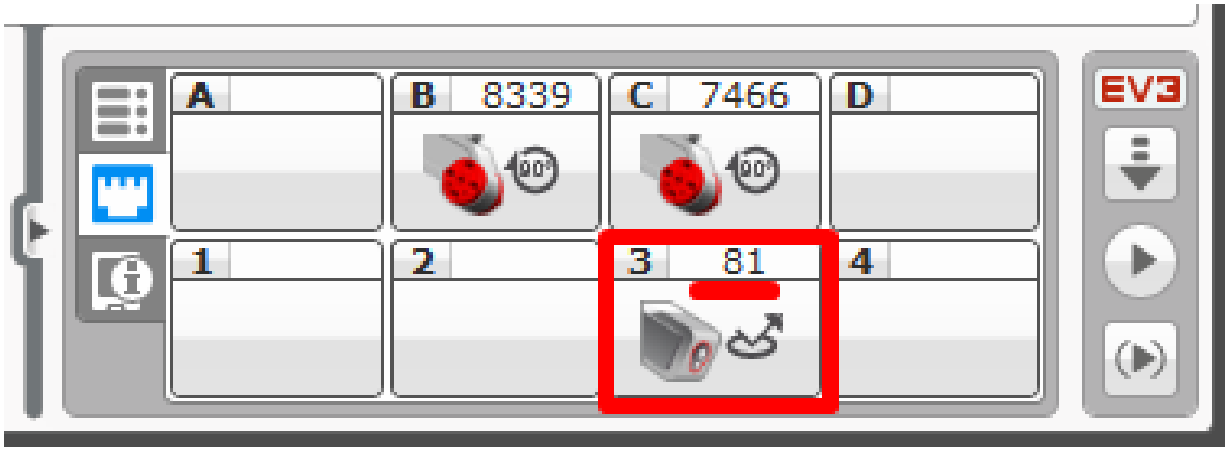



Figura 19.5: Valor medido na parte clara

Quando utilizamos o `Colour Sensor` no modo `Reflected Light Intensity`, o sensor emite um feixe de luz de cor vermelha. Pois bem, nesse modo o sensor mede a quantidade de luz que foi refletida para ele. Uma superfície escura (como a linha negra da pista, por exemplo) reflete pouca luz, já uma superfície clara refletirá bastante luz. As superfícies escuras tendem a um valor próximo de 0 e, as claras, a valores próximos de 100. Logo, haverá uma disparidade bem

grande entre o valor lido em uma superfície clara e uma superfície escura.

Conforme você pôde observar nas imagens, o sensor mediu o valor `8` para a linha negra e `81` para a superfície branca em volta da linha. Veja, o valor medido na linha negra é muito baixo em relação à superfície branca em volta da pista.

Agora que temos os valores, devemos criar uma sequência de código que possibilite o veículo a seguir o traçado negro.

## 19.1 O código

Como o valor medido na parte negra foi `8` e na parte clara foi `81` , podemos definir no parâmetro `Threshold Value` (do `Switch` , que veremos a seguir) de `50` para trabalhar com as duas situações no `Switch` . Ora, você deve estar pensando: "por que `50` se o valor medido na parte escura é muito inferior a isso?". As medições do sensor podem variar e, por isso, o valor `50` garante uma boa segurança, e além disso, satisfaz nossa situação perfeitamente.

Durante a execução do programa podem ocorrer variações no valor lido pelo sensor em alguns trechos da pista em relação à quantidade de luz refletida. Por esse motivo, devemos adotar um valor de comparação com uma boa margem de segurança. Deste modo, mesmo que ocorra uma variação na parte clara e na parte escura, nosso programa não apresentará problemas durante sua execução.

O valor `50` nos é satisfatório, pois ele é um valor médio (relembrando: a escala vai de 0 a 100), e apresenta uma grande distância do valor `8` e de `81` . Portanto, mesmo que aconteçam variações bruscas no valor lido pelo sensor, a lógica do programa não será afetada.

Contudo, outros valores podem ser utilizados como valor limiar, desde que atendam às condições já citadas.

Por exemplo, se escolhêssemos trabalhar com o `Switch` executando uma sequência para um valor maior `8` , qualquer pequena variação faria com o nosso robô não executasse as tarefas conforme o desejado.

O primeiro bloco que vamos adicionar (após o bloco `Start` , claro) é o bloco `Loop` , configurado em `Unlimited` , para que o programa execute as ações infinitamente.

A seguir, vamos inserir o `Switch` , que será responsável pela parte lógica do robô. Nele, vamos realizar as configurações que permitirão que o robô siga o traçado negro. Para isso, vamos configurá-lo da seguinte forma: `Colour Sensor - Compare - Reflected Light Intensity` . Com isso, dizemos ao `Switch` que queremos comparar o valor da intensidade da luz refletida. O próximo passo é escolher o tipo de comparação, o que é feito em `Compare Type` . Vamos escolher a comparação `Greater Than` (Maior que), representada pelo número 2 e, no `Threshold Value` , vamos colocar o número que tomamos como referência: o `50` . Assim, concluímos mais uma etapa do código.

Quando o robô atingir a parte clara (luminosidade maior que `50` ) uma determinada sequência de código será executada.

Logo, vamos inserir nessa parte do `Switch` dois blocos `Large Motor` , o bloco B com potência de valor igual a `10` , e o C, ajustado com um valor de `50` .

Quando o robô atingir a parte escura (luminosidade menor que `50` ) outra sequência de código entrará em ação.

Devemos seguir o mesmo procedimento que fizemos no `Case` anterior, porém, vamos configurar as potências do motor de forma contrária. Sendo assim, o motor B passará a ter uma potência igual a `50` e o C, uma potência igual a `10` .

Seguindo todos esses passos, tem-se como resultado o seguinte código:

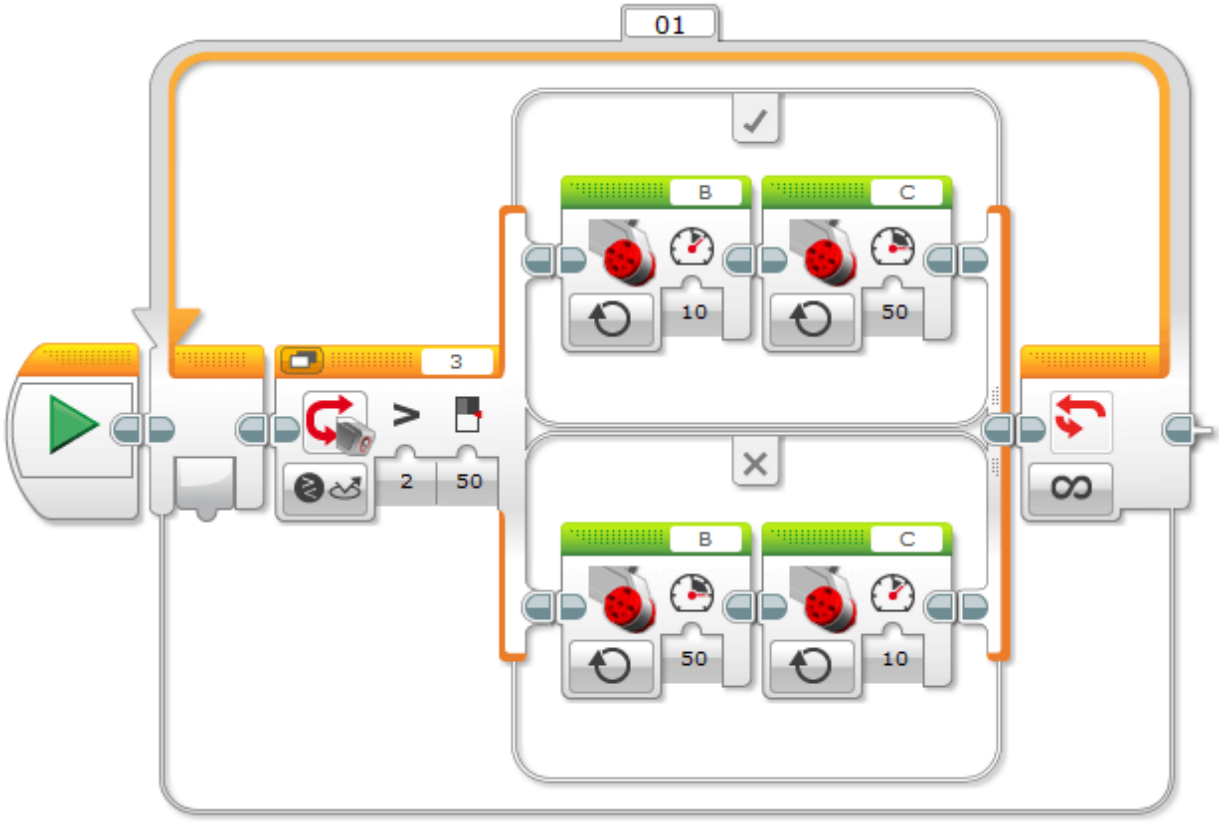



Figura 19.6: Seguidor de Linha: Compare Reflected Light Intensity

## Entendendo o movimento do robô

Com a escolha de dois blocos `Large Motor` ajustados com potências diferentes, o programa funciona da seguinte forma: o robô realizará um movimento semelhante a um zigue-zague. Quando identificar a cor preta (luminosidade menor que `50` , logo, ele executará o `Case - False` do `Switch` ) ele desacelera o motor C, e acelera o motor B. Quando identificar a cor branca da pista (luminosidade maior que `50` ) ele desacelera o motor B e acelera o motor C. Como temos um motor do lado esquerdo e outro do lado direito com valores distintos de potências, isso resultará em um movimento de zigue-zague.

Esse foi apenas o primeiro código, a seguir serão apresentadas formas mais desafiadoras. Por meio das variações do código que acabou de ser apresentado, você aprenderá a criar códigos cada vez mais completos e sofisticados. Depois de estudar as variações, você poderá testar seus conhecimentos com os exercícios propostos. Ao final, ou caso tenha alguma dúvida, você pode consultar as resoluções do exercício.

## 19.2 Variações do Robô Seguidor de Linha

A partir do código do Robô Seguidor de Linha, podemos desenvolver algumas variações que alteram o modo de funcionamento do programa, mas preservam sua lógica principal.

A ideia desta seção é apresentar formas alternativas de solução para o problema. Dessa forma, além de conhecer e aprender formas diferentes de solução, você conhecerá mais funcionalidades acerca dos blocos de programação. Porém, lembre-se: a melhor solução é aquela que apresentar melhor resultado. Em determinadas situações, uma solução pode ser melhor que outra, às vezes não. O principal objetivo é que você conheça essas formas e saiba utilizá-las quando necessárias.

### Utilizando o bloco Colour Sensor no modo Compare - Colour

A principal vantagem deste programa é a sua comodidade, pois é muito mais fácil utilizar um sensor de cor no modo cor, do que no modo de luz (como demonstrado no primeiro programa). No modo cor, pode-se escolher uma cor para ser comparada, ou várias cores, o que torna o trabalho muito mais fácil e automático.

A única alteração necessária neste programa é no modo de operação do sensor de cor.

A seguir está a imagem representando com mais detalhes a configuração do sensor para o modo cor.

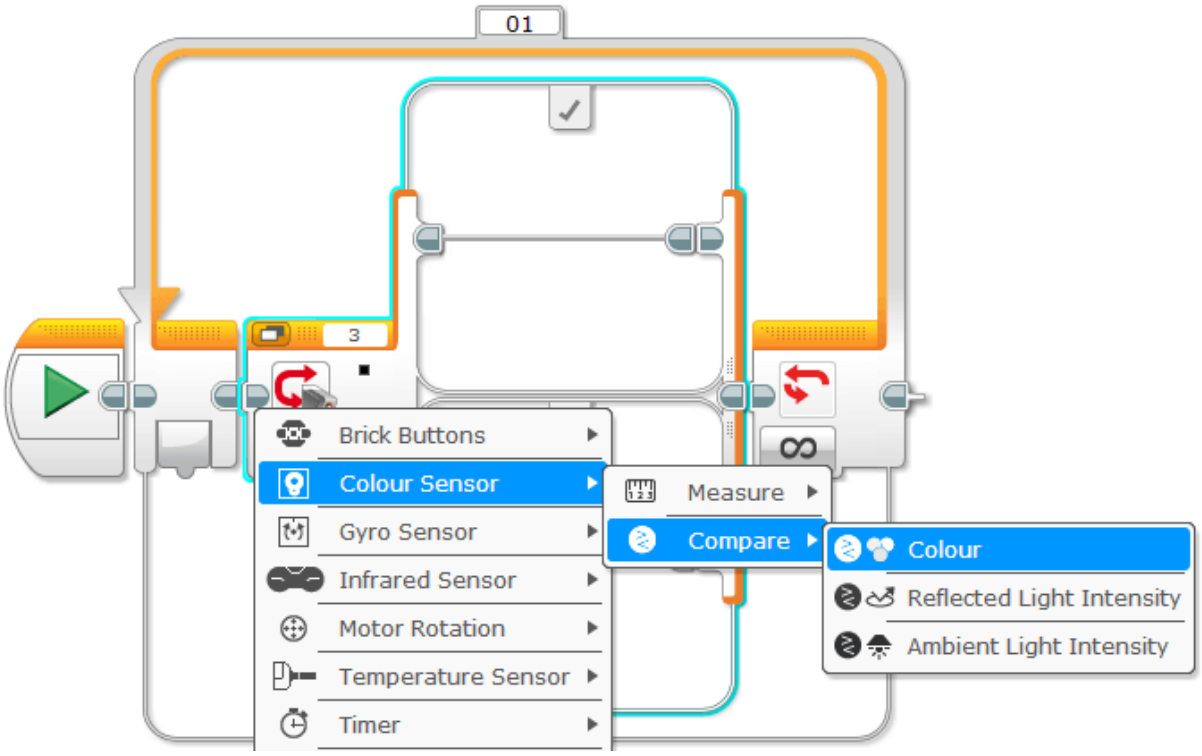



Figura 19.7: Configurando o Case (Caso) do Switch

No `Switch` escolhe-se a opção `Colour Sensor` , em seguida, a opção `Compare` e, por fim, `Colour` .

O parâmetro `Set of Colours` é onde definimos a cor que será comparada, conforme ilustra a figura a seguir.

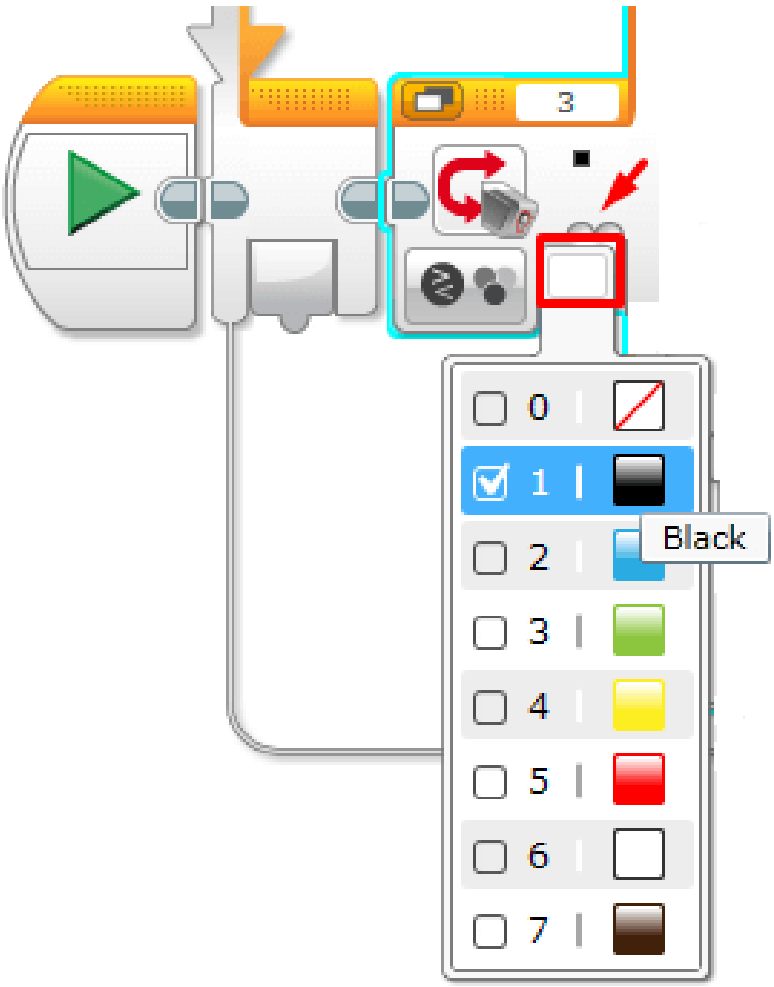



Figura 19.8: Seguidor de Linha: Compare - Colour - Set of Colours

Aqui está o programa completo:

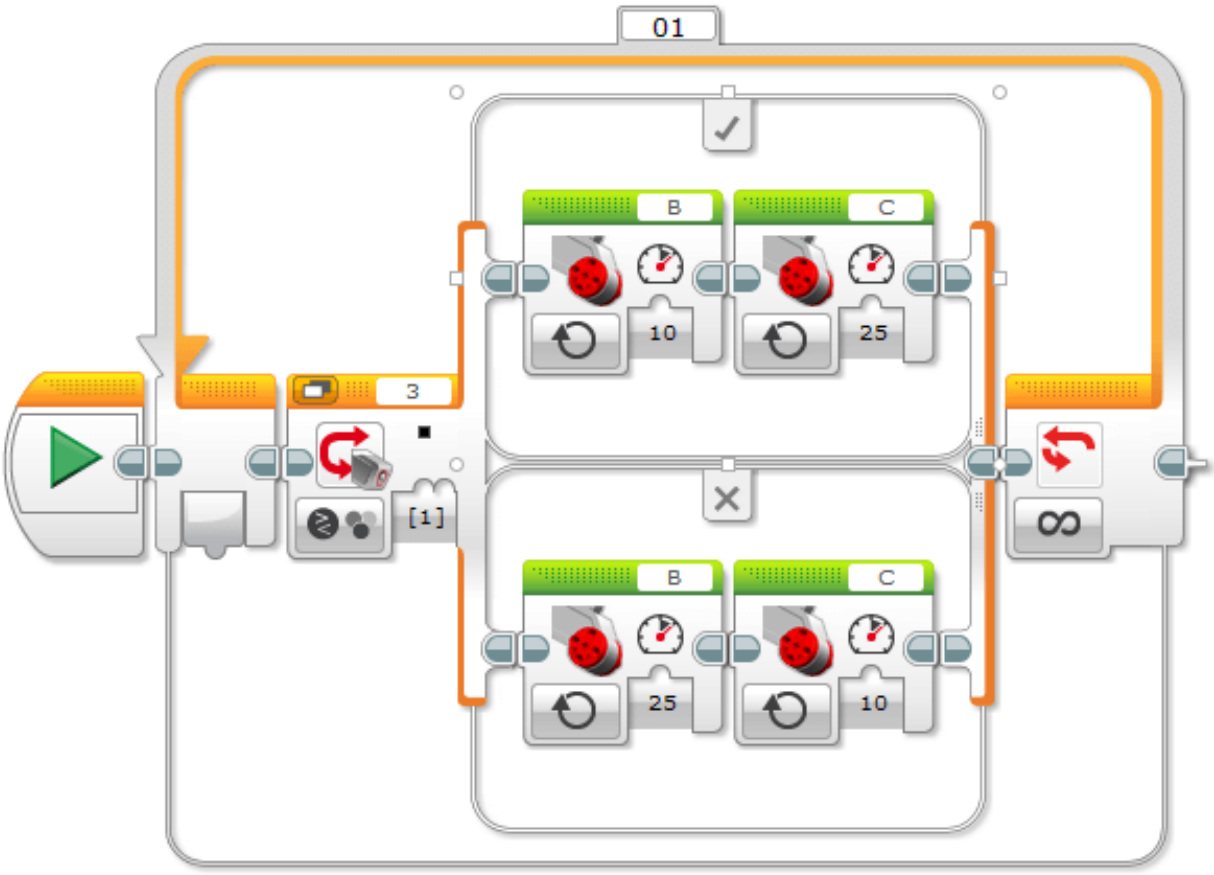



Figura 19.9: Seguidor de Linha: Compare - Colour

## Utilizando o modo Measure - Colour

Uma outra forma de utilizar o sensor de cor no modo cor poderia ser com o modo `Measure` (medir). Após selecionar o modo, basta escolher a cor que será medida em cada `Case` . O primeiro caso do `Switch` já vem com a cor `Black` (preto) selecionada, dessa forma, basta alterarmos a cor do segundo caso para `White` (branco).

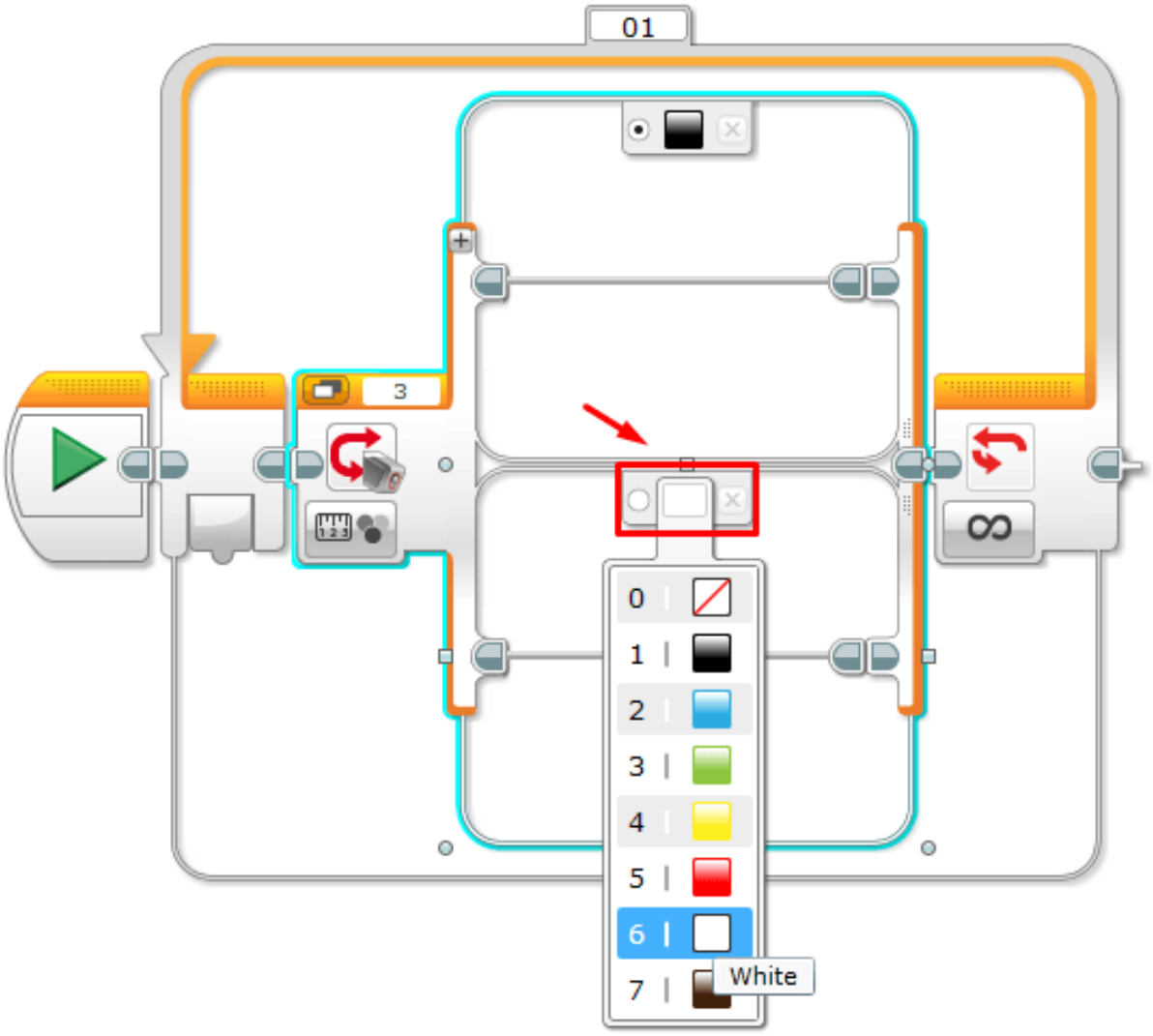



Figura 19.10: Configurando o Switch para o modo Colour

Observe como ficou o programa:

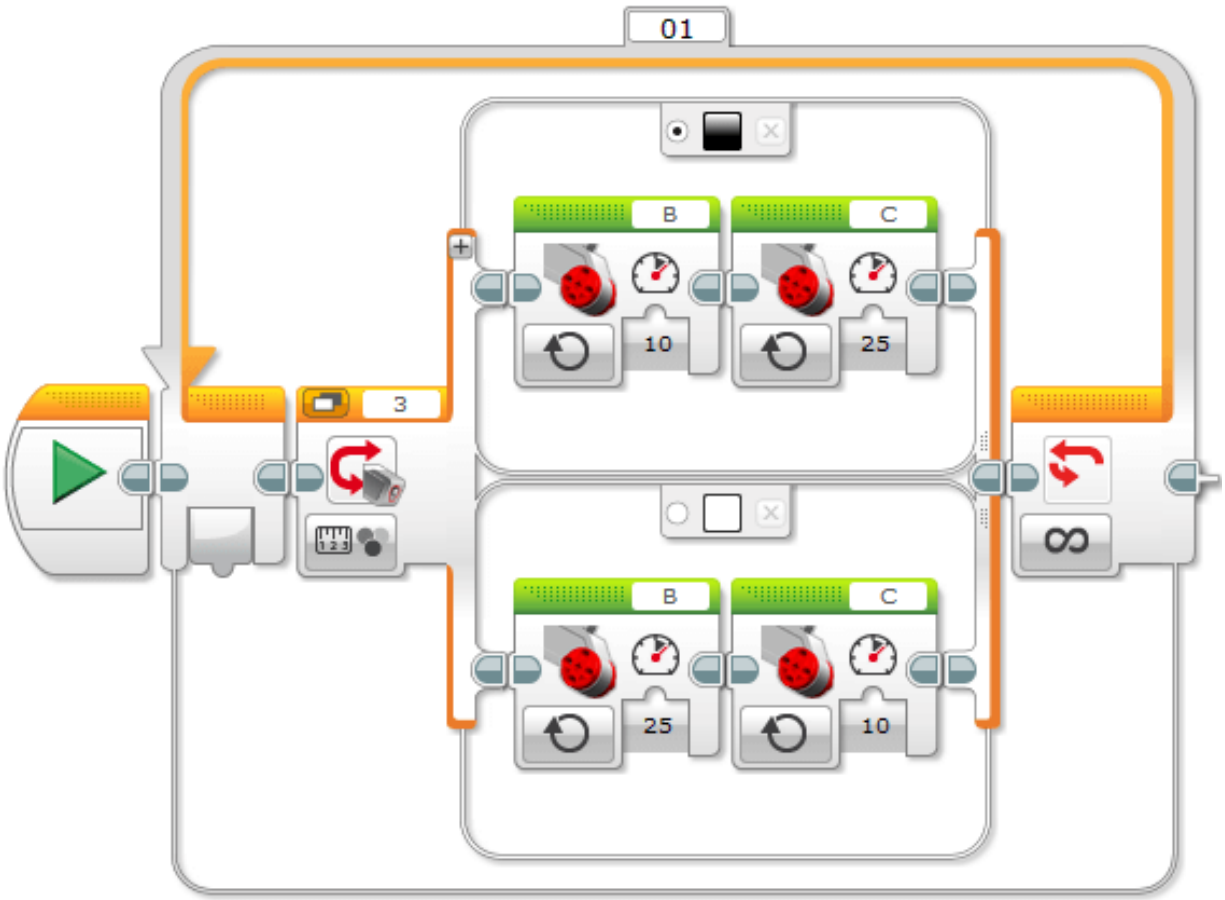



Figura 19.11: Seguidor de Linha: Measure - Colour

Essas são algumas das possibilidades de solucionar o problema, com a mesma lógica, porém, com uma diferença apenas quanto ao modo de funcionamento do sensor de cor. Agora, quando você for utilizar o sensor de cor, já saberá quais são suas possibilidades e assim poderá utilizá-lo da melhor forma possível.

**Mais alguns blocos!**

Antes de ir para os exercícios, serão apresentados alguns blocos que você utilizará para resolver os exercícios. Desta forma, você terá um ponto de partida para resolvê-los.

## 19.3 O bloco Math

Figura 19.12: Ícone do bloco Math

Este bloco é utilizado para realizar operações matemáticas, por exemplo, é possível utilizá-lo para realizar um incremento em uma variável, entre muitas possibilidades.

O bloco `Math` na maioria dos modos é composto por 2 entradas e uma saída: A, B e `Result` . A e B são entradas, pode-se definir um valor ou enviar um valor para eles por meio de um `Data wire` . Na saída `Result` pode-se enviar o resultado da operação para outro bloco.

A seguir estão os modos de operação:

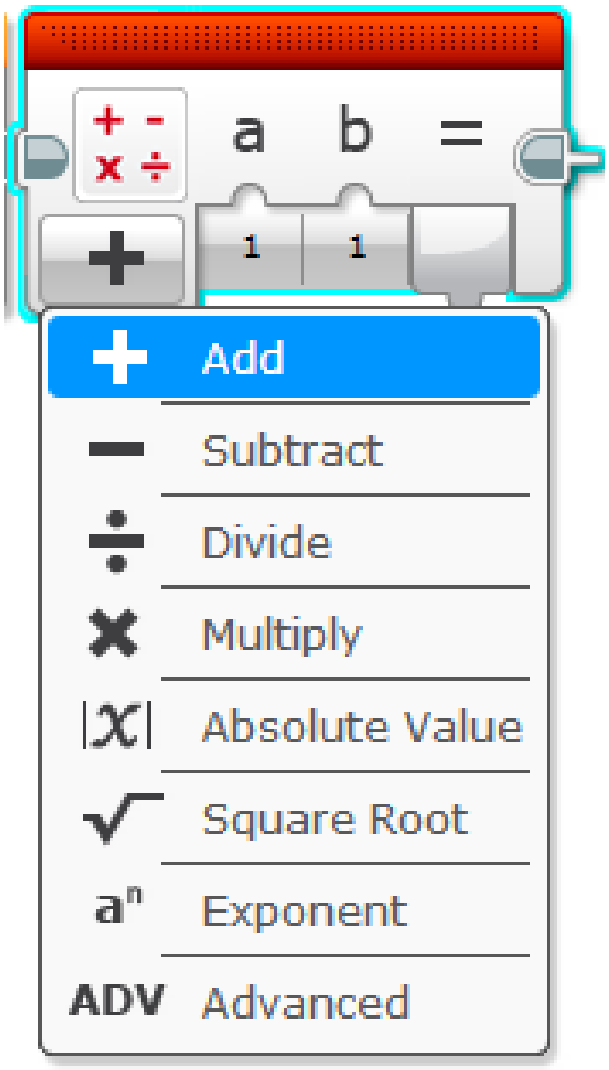



Figura 19.13: Modos do bloco Math

- `Add` : realiza soma.
- `Subtract` : realiza subtração.
- `Divide` : realiza divisão.
- `Multiply` : realiza multiplicação.
- `Absolute Value` : define um valor absoluto.
- `Square Root` : realiza a radiciação (tira a raiz quadrada).
- `Exponent` : eleva a um expoente. Este modo não apesenta o parâmetro “B”; no lugar dele está o “n”, onde fica o número do expoente ao qual vai elevar a base (que é a entrada A).

- `Advanced` : usado para expressões numéricas. Possui quatro entradas: A, B, C e D. No valor do bloco, pode-se escolher dentre várias operações, muitas das quais não possuem modo próprio no bloco - elas são operações mais complexas, ou seja, “advanced”.

## 19.4 O bloco Compare

Figura 19.14: Ícone do bloco Compare

Bloco responsável por fazer comparações. Ele compara dois parâmetros de acordo com o modo selecionado e retorna um resultado. Pode-se utilizar o `Data wires` vindo de uma variável para uma entrada e atribuir um valor a ser comparado em outra, ou ainda comparar duas variáveis.

Possui os seguintes modos:

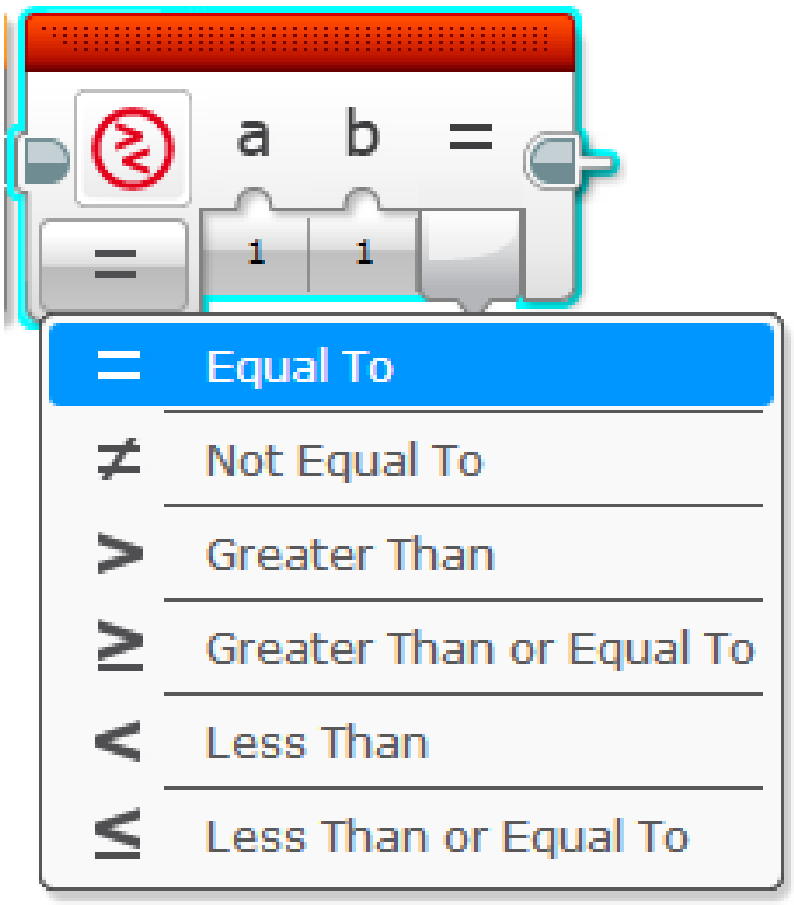



Figura 19.15: Modos do bloco Compare

- `Equal To` : igual a
- `Not Equal To` : diferente de
- `Greater Than` : maior que
- `Greater Than or Equal To` : maior ou igual a
- `Less Than` : menor que
- `Less Than or Equal To` : menor ou igual a

## 19.5 Exercícios

**Agora, vamos às atividades!**

1. Elabore um código que permita ao veículo seguir uma linha negra por oito segundos. Após este tempo, o robô emitirá um som e dará meia volta, para continuar a seguir a linha escura no sentido contrário por mais oito segundos. Esta sequência de movimentos deverá se repetir por: a.Ininterruptamente; b.Três vezes.

2. Refaça o item b do problema 1 utilizando, para a contagem, variáveis e condições lógicas.

3. Refaça o problema 1 utilizando, para a análise de intensidade luminosa, variáveis e condições lógicas.

**DICA**

Conforme foi possível observar na foto do Robô Seguidor de Linha, o sensor de cor está localizado aproximadamente no centro do robô. Isso melhora muito a eficácia do robô, pois desta forma, quando ele realizar o giro, o sensor estará muito próximo à linha negra, facilitando e executando com mais perfeição o trajeto.

Para resolver o item b do exercício 1, lembre-se de que o Loop oferece um modo que torna isto bem mais fácil.

## 19.6 Resoluções

**1. a)**

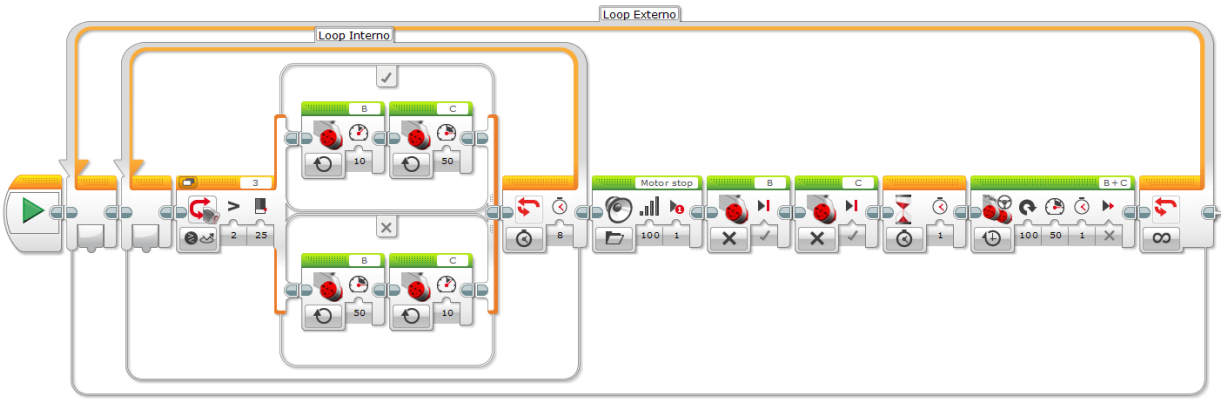



Figura 19.16: Solução do item "a" da questão - Código completo

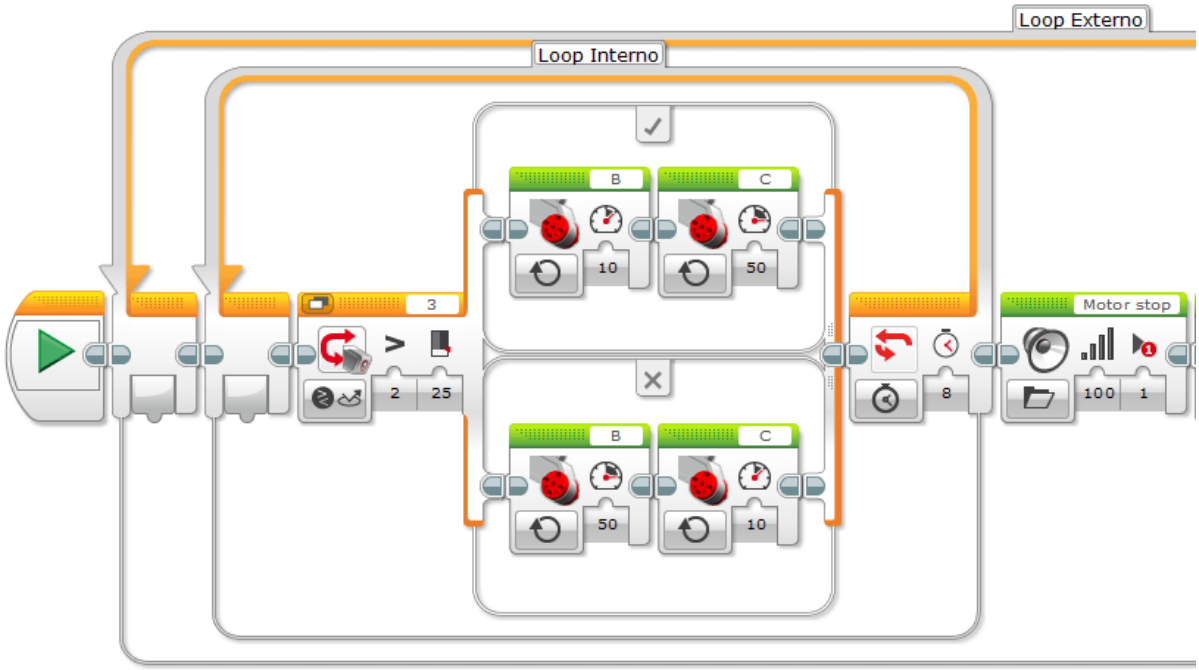



Figura 19.17: Solução do item "a" da questão - Detalhe 1

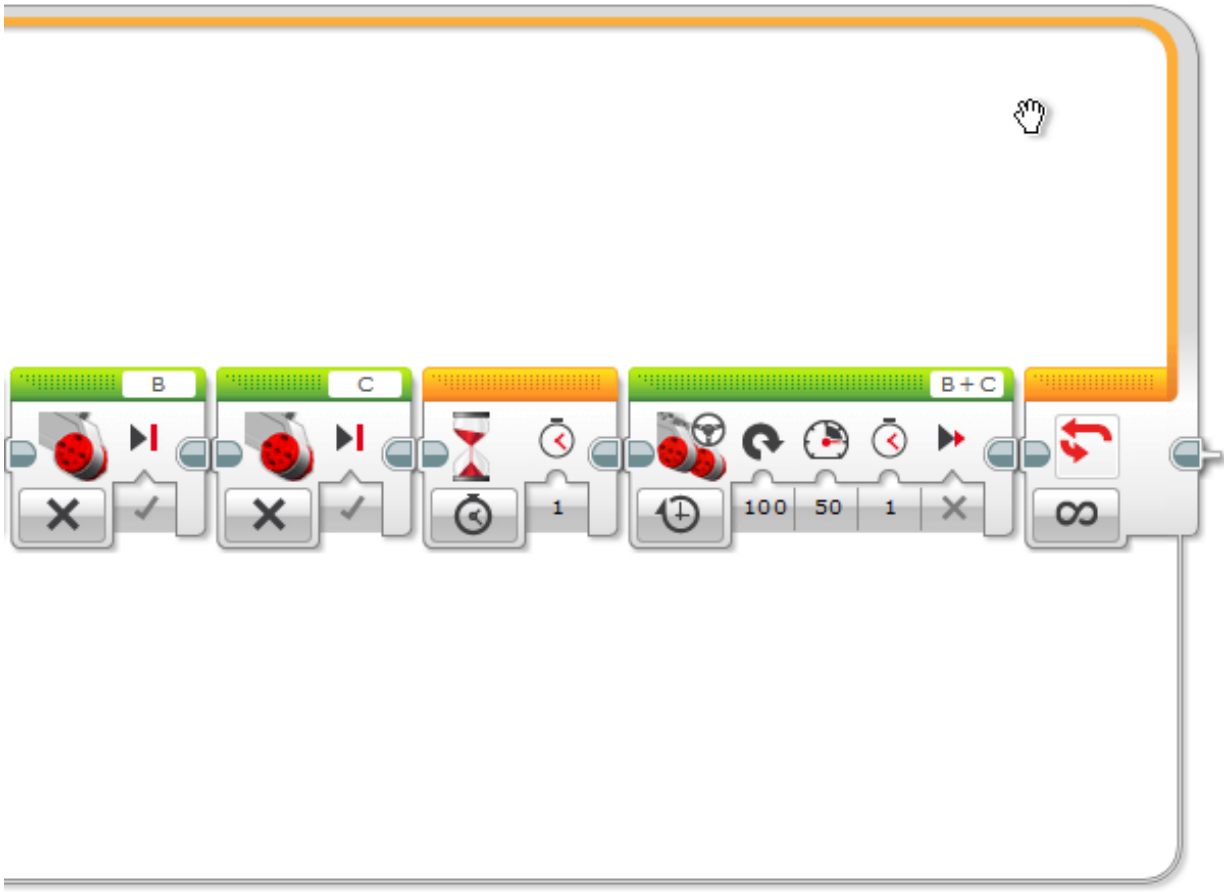



Figura 19.18: Solução do item "a" da questão - Detalhe 2

Este código apresenta o incremento de um bloco responsável pela volta (o `Move Steering` ), um bloco `Sound` , e mais um `Loop` , que agora realiza o movimento por oito segundos. Neste código, como é perceptível, temos dois `Loops` em sua composição.

O `Loop` interno contém os blocos que executam o movimento do robô que é responsável por seguir a linha, e é executado durante oito segundos, para isso basta configurar o `Loop` no modo `Time Indicator` e alterar a sua duração. Após esses oitos segundos, o `Loop` se encerra.

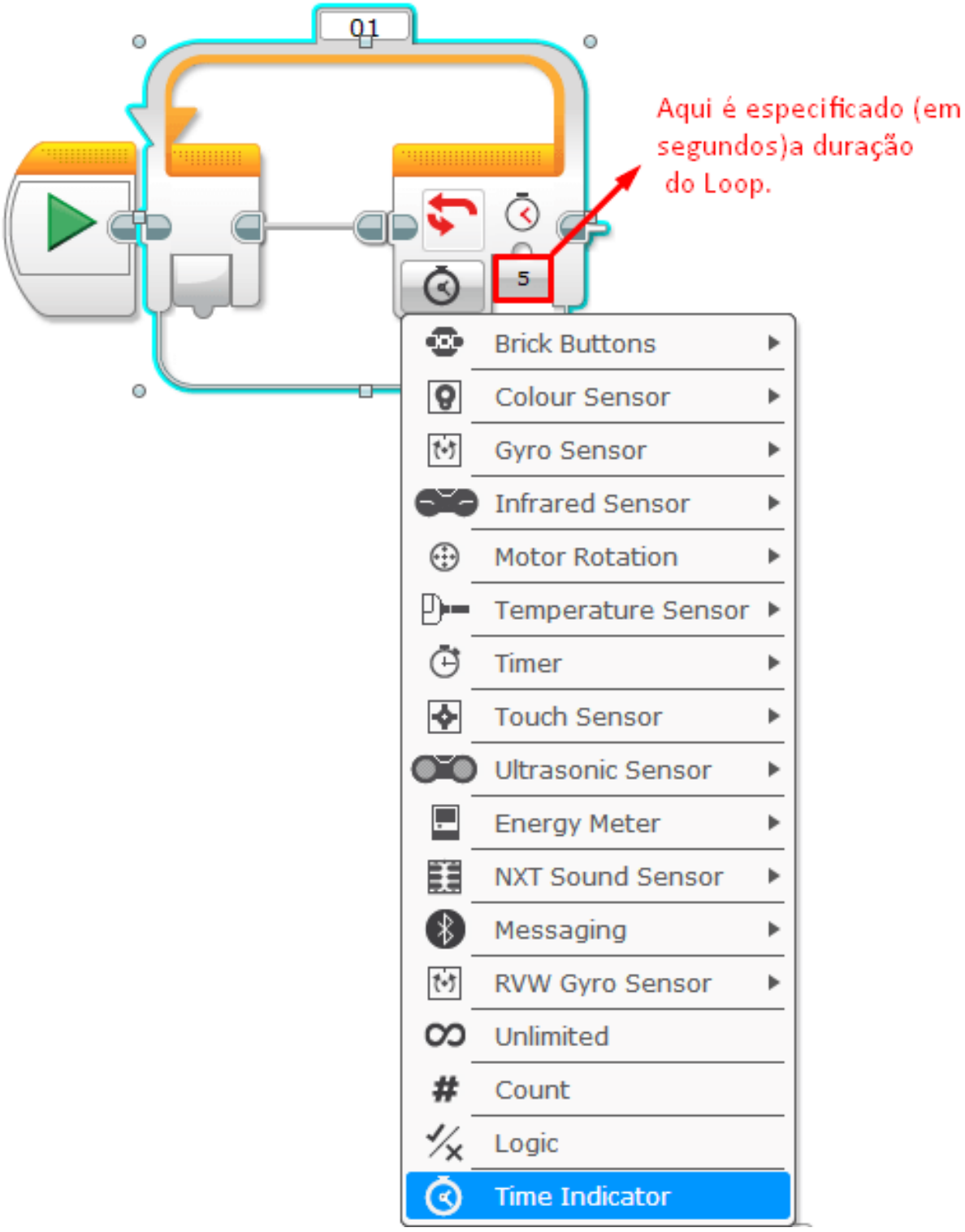



Figura 19.19: Solução do item "a" da questão - Configurando o Loop

Depois, tem-se o `Loop` externo, que é de duração infinita e que contém o bloco `Move Steering` , responsável por executar o movimento de meia volta do robô.

Configura-se o bloco `Move Steering` para funcionar por segundos, em seguida, ajusta-se o `Steering` , que nesse caso recebeu o valor 100. O parâmetro `Seconds` permanece com 1 segundo; no parâmetro `Brake at End` , foi selecionada a opção `Coast` , ela faz com que o desligamento dos motores seja mais suave.

**1. b)**

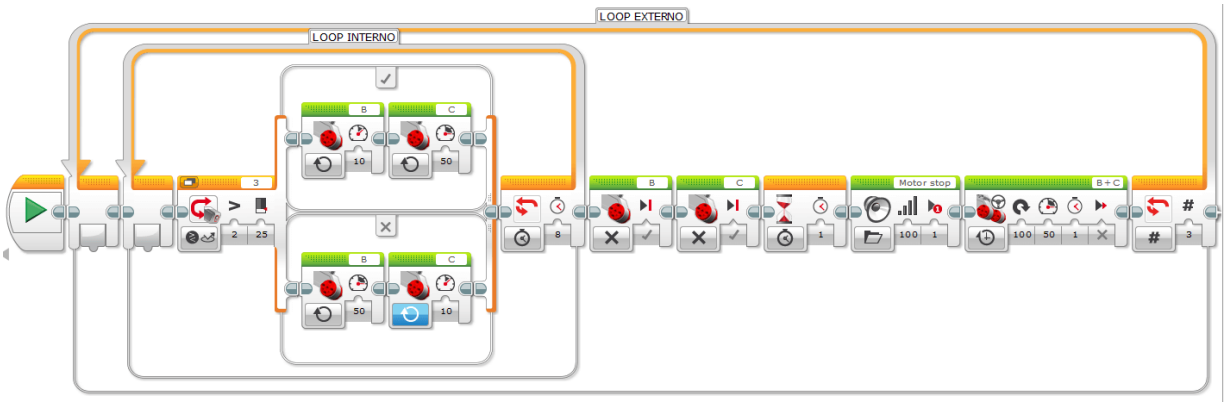



Figura 19.20: Solução do item "b" da questão

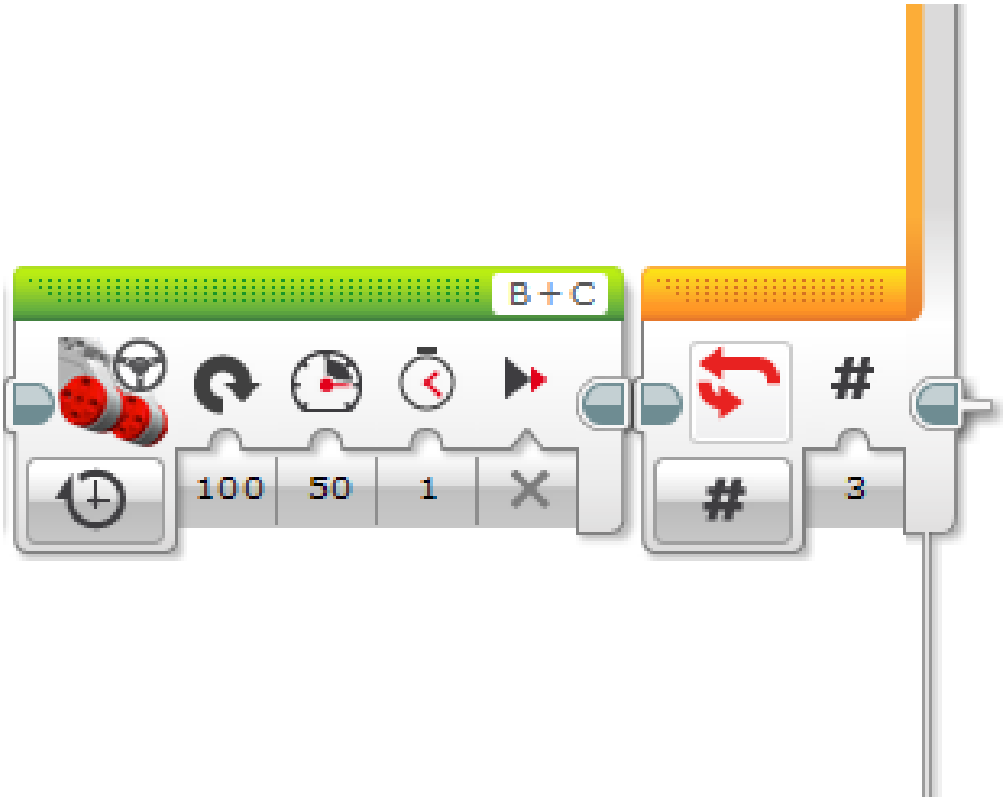



Figura 19.21: Solução do item "b" da questão - Detalhe - Parte final do código

Para que o robô execute o movimento por um número definido de vezes, basta alterar o `Loop` . Usa-se a configuração `Count` . Com ela, define-se o número de vezes que o `Loop` é executado.

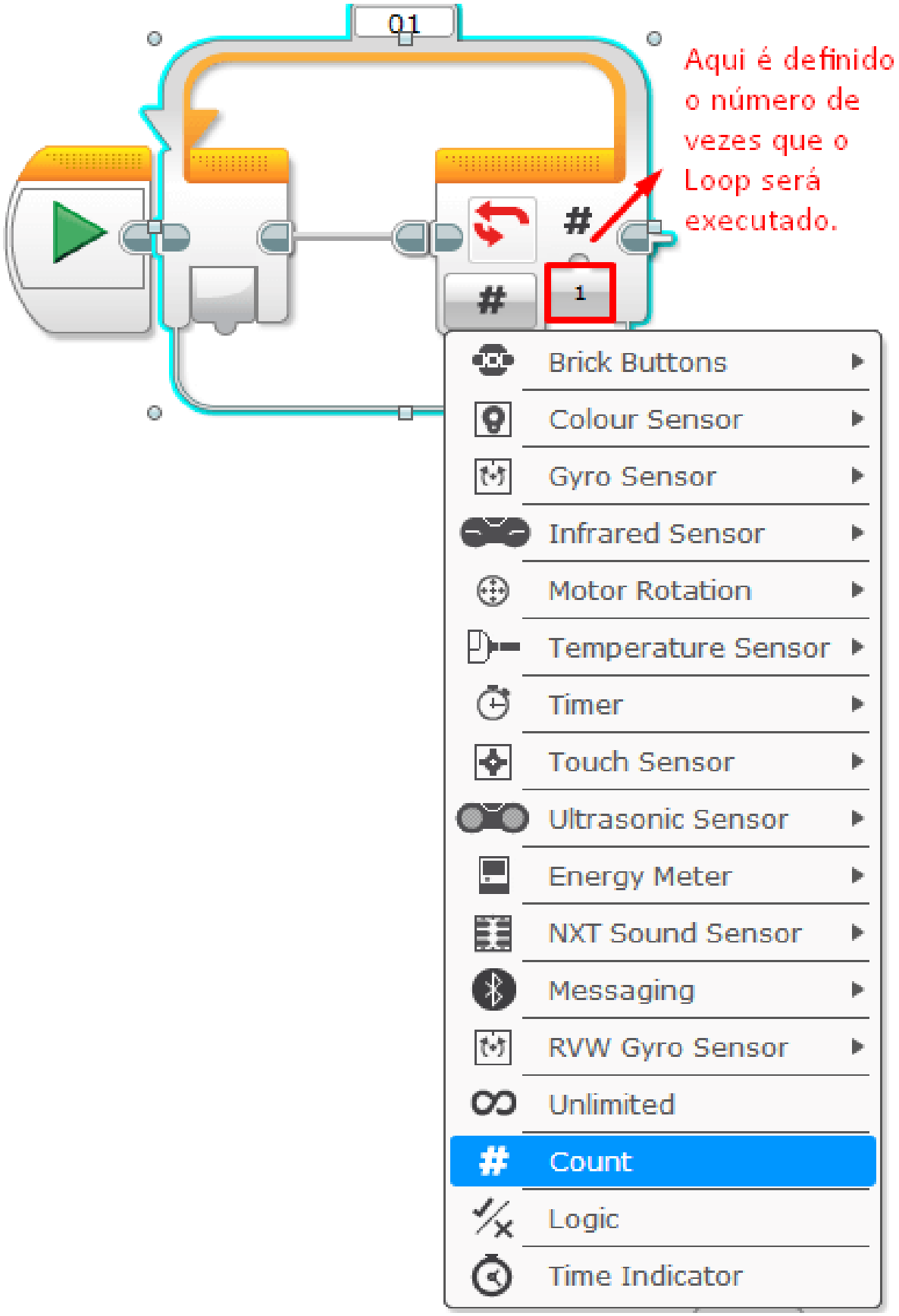

01
#
#
1
Aqui é definido o número de vezes que o Loop será executado.
Brick Buttons
Colour Sensor
Gyro Sensor
Infrared Sensor
Motor Rotation
Temperature Sensor
Timer
Touch Sensor
Ultrasonic Sensor
Energy Meter
NXT Sound Sensor
Messaging
RVW Gyro Sensor
Unlimited
Count
Logic
Time Indicator

Figura 19.22: Solução do item "b" da questão - Configurando o Loop

**2.**

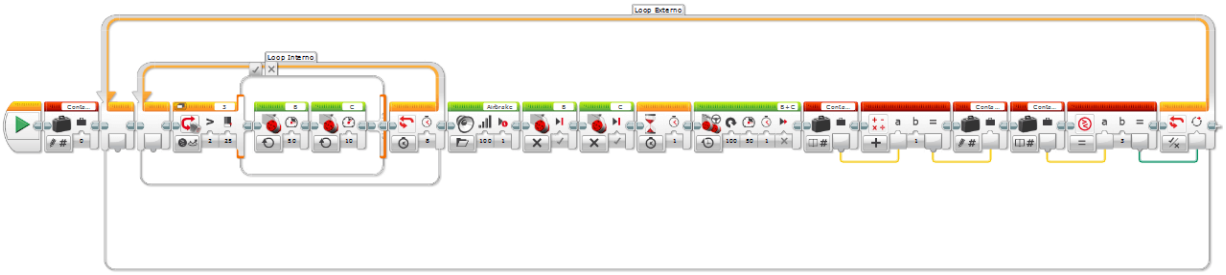


Figura 19.23: Solução da questão 2

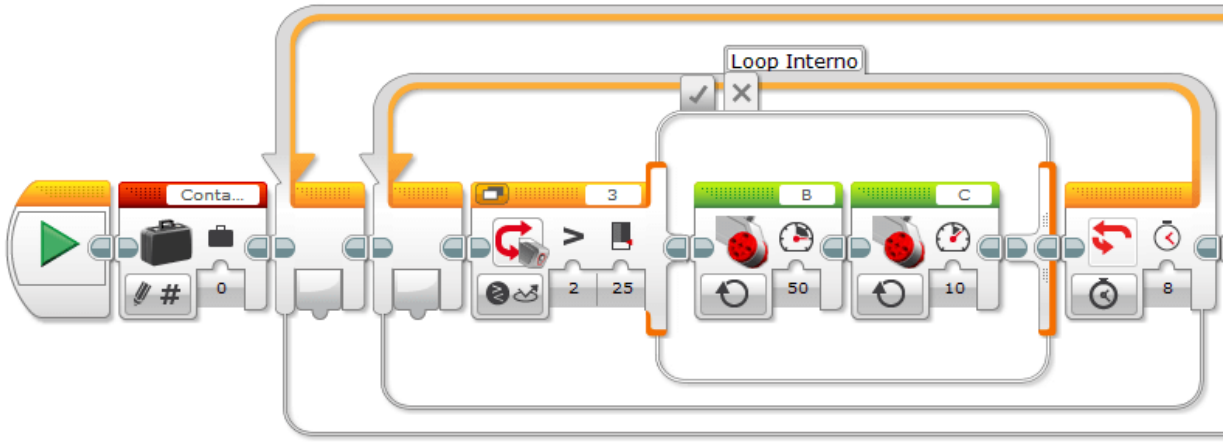



Figura 19.24: Solução da questão 2 - Detalhe 1

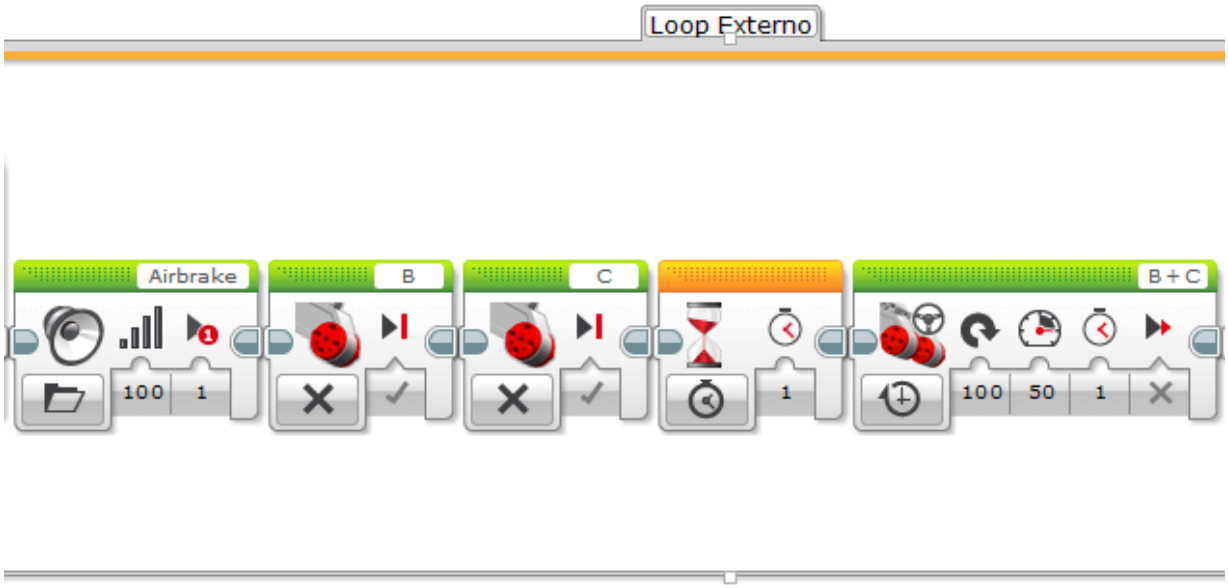



Figura 19.25: Solução da questão 2 - Detalhe 2

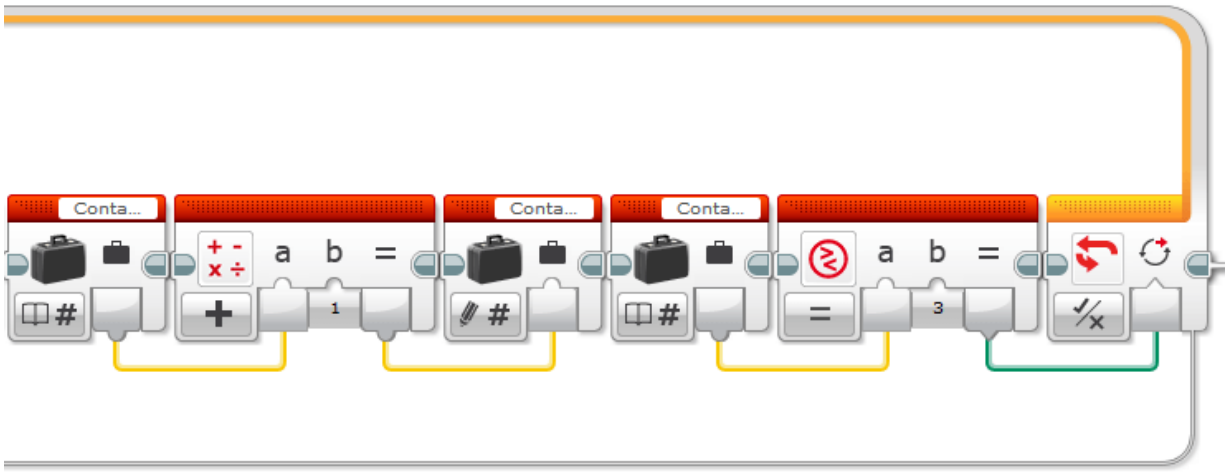



Figura 19.26: Solução da questão 2 - Detalhe 3

O item “b” da primeira questão poderia ser resolvido com o uso de variáveis, como faremos neste exercício, porém o uso de `Count` facilita o trabalho e otimiza o tempo. Com este exercício, vamos trabalhar mais uma vez com a utilizações de variáveis, que são de grande auxílio e extremamente úteis e importantes em diversas ocasiões.

O primeiro passo é adicionar uma variável no início do programa e antes do `Loop` . Essa variável será numérica, pois nela guardamos o número de vezes que o `Loop` interno foi executado. Nessa variável escrevemos o valor `0` , pois sempre quando o programa for iniciado a contagem começará em 0. A variável deve ser inicializada antes do `Loop` pois o programa vai lê-la; caso a variável estivesse dentro do `Loop` a contagem seria impossível, pois a variável sempre iria voltar a possuir valor zero.

O próximo passo consiste em criar a parte matemática do código. Conforme explicado, devemos adicionar uma unidade à variável `Contagem` a cada vez que o robô executa as ações. A variável deve ficar após o bloco `Move Steering` , pois, dessa forma, a cada giro o robô contabiliza uma execução. A contagem só é realizada após a execução do movimento. Para isso, utilizamos um bloco matemático, o bloco `Math` , contido na aba `Data Operations` .

Figura 19.27: Solução da questão 2 - Adicionando variáveis

Configuramos o bloco com a opção `Add` , ou seja, adicionar. Para isso, escrevemos o número `1` em `a` , e o resultado dessa operação deve ser fornecido à variável `Contagem` . Para isso, vamos realizar uma ligação, que é feita pelo fio amarelo, conforme mostrado na figura. Antes de conectar o fio, precisamos nos certificar de que a variável está configurada no modo `Write` (escrever), pois caso contrário não será possível realizar a ligação.

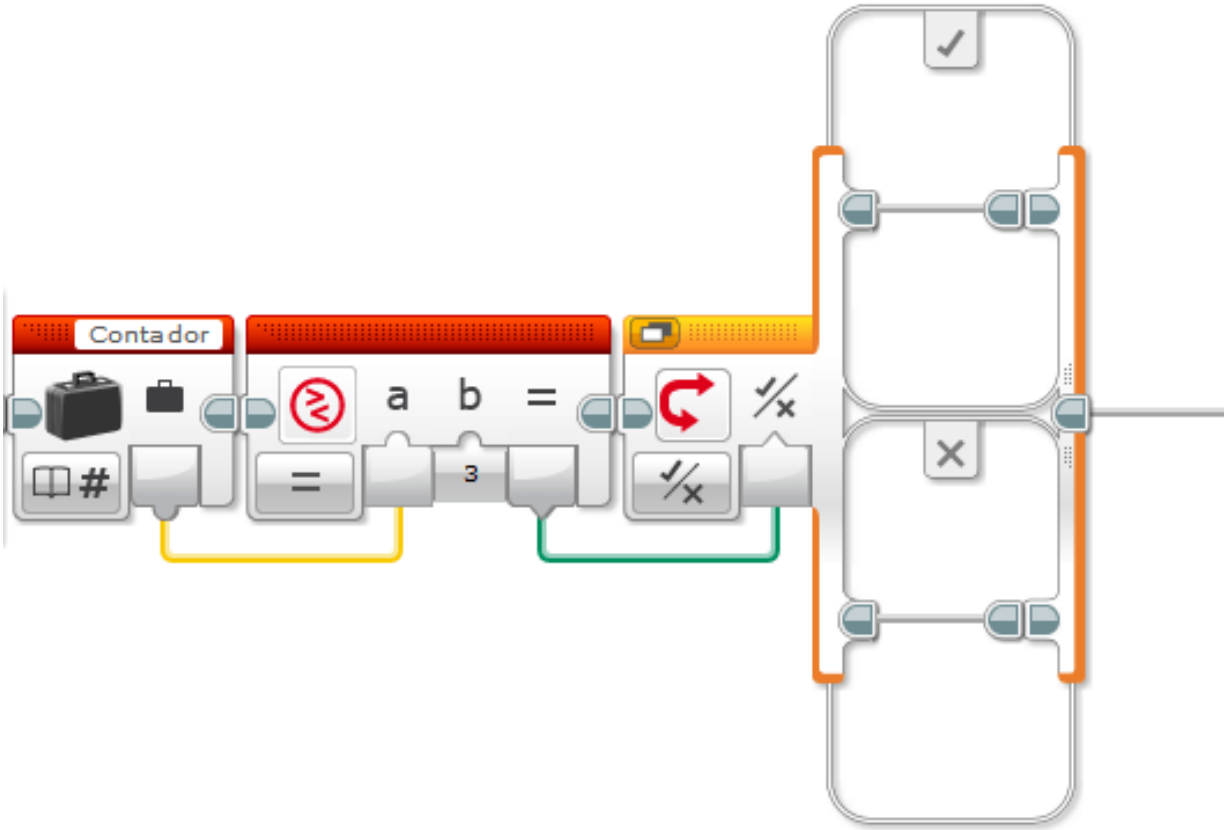



Figura 19.28: Solução da questão 2 - Adicionando variáveis

Aqui está a parte final do nosso programa, a parte lógica. Nessa parte, será feita a lógica para que, quando o robô realizar os 3 giros, o programa se encerre.

Duas coisas terão que ser feitas para a construção da lógica. A primeira será utilizar o bloco matemático `Compare` , com ele será comparado o valor da variável. O bloco `Compare` é responsável por comparar se o valor da variável é igual a 3, conforme solicitado no enunciado. Para isso adicionamos a variável `Contagem` configurada no modo `Read` , o modo de leitura, em seguida, adicionamos o bloco `Compare` , que deve ser configurado no modo `Equal To` (igual a). Na entrada “a” ligamos o fio vindo da variável, e em “b” escrevemos o valor a ser comparado (o número 3). Isso significa que, quando a variável possuir o valor igual a 3, o resultado fornecido pela comparação será verdadeiro. Esse valor obtido através do bloco

`Compare` será enviado ao `Loop` , que foi configurado para o modo `Logic` , conforme veremos.

Após realizar isso, é necessário configurar o `Loop` para o modo `Logic` , e ligar o resultado do bloco `Compare` a ele, isso é feito conectando a saída `Result` do bloco `Compare` à entrada `Until True` .

Figura 19.29: Solução da questão 2 - Construindo a parte final do código

**3.**

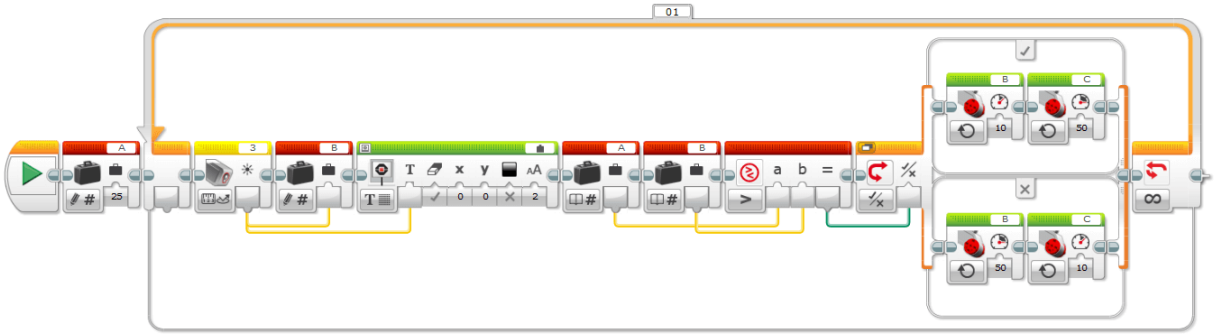


Figura 19.30: Solução da questão 3

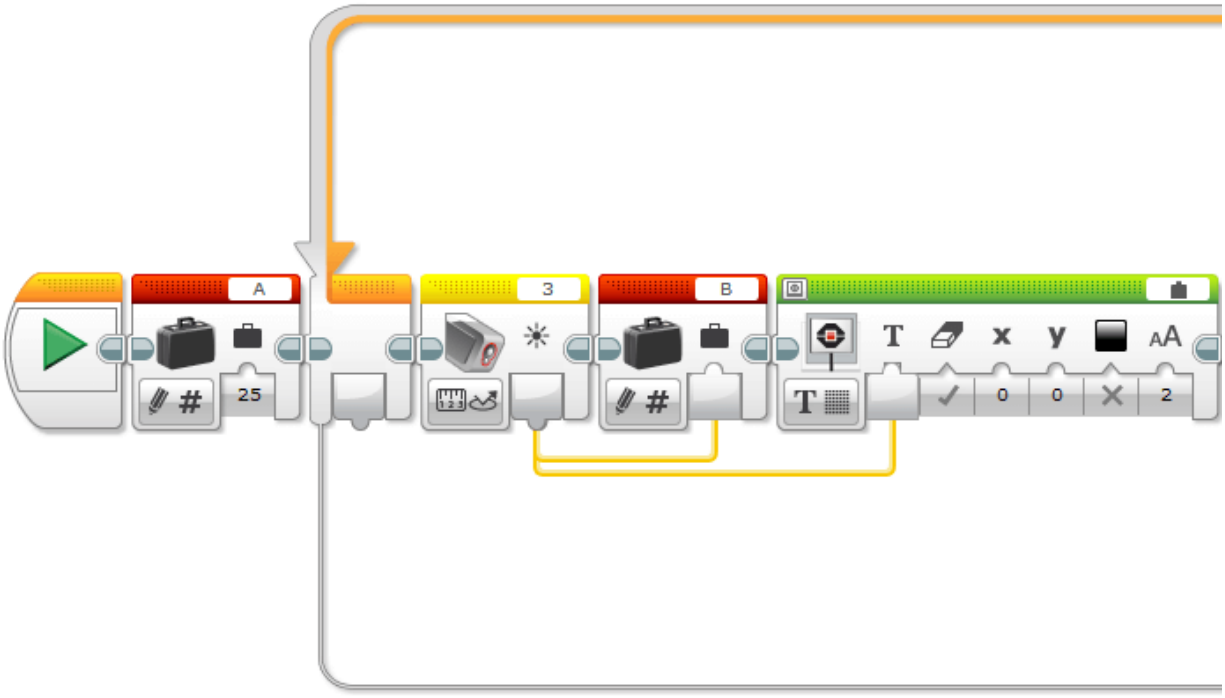



Figura 19.31: Solução da questão 3 - Detalhe 1

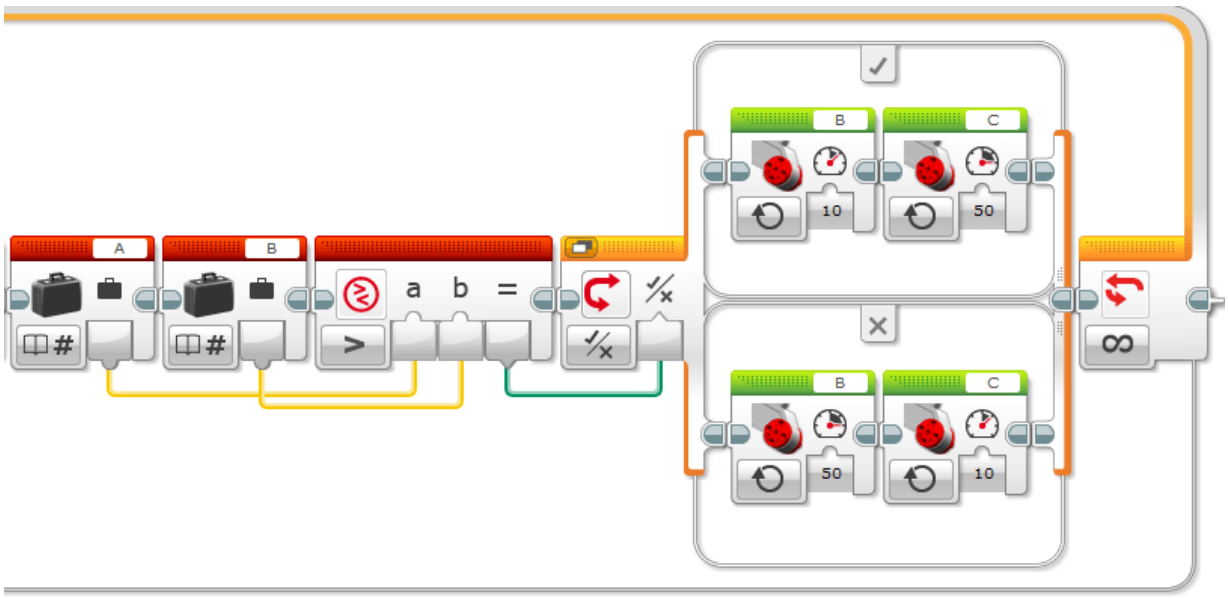



Figura 19.32: Solução da questão 3 - Detalhe 2

Para este programa vamos usar duas variáveis, ou uma variável e uma constante. Primeiro, utilizamos uma variável antes do Loop , que foi chamada de A , nela escrevemos o valor 25 , seria possível

também a utilização de uma constante, já que este valor é fixo e não sofre alteração. Este valor servirá como valor limiar ( `Threshold Value` ).

> **CONSTANTES**
>
> Utilizamos constantes quando queremos registrar um valor fixo. Por exemplo, para que uma comparação seja realizada, precisamos definir um valor limiar. A partir desse valor limiar serão feitas as comparações. No programa, optou-se por utilizar uma variável, mas ela é dedicada exclusivamente para guardar o valor limiar, e não sofre nenhuma modificação. Porém, como mencionado no início, poderíamos ter utilizado uma constante.

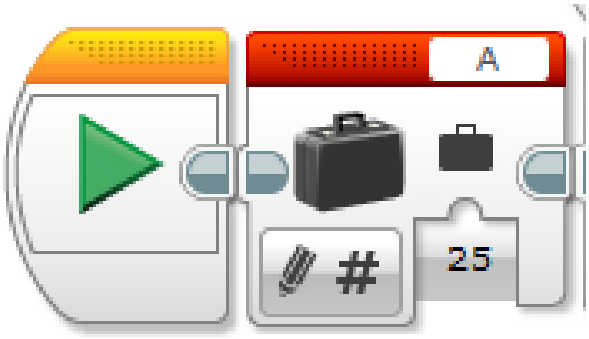



Figura 19.33: Solução da questão 3 - Adicionando variáveis

Agora vem o `Loop` . Para que se possa realizar uma comparação, é necessário "pegar" o valor lido pelo sensor de luz e guardá-lo em uma variável. Isso será feito na variável `B` , e esse valor será enviado ao bloco `Display` também, para que seja exibido durante a execução do programa.

Para que seja realizada a leitura, devemos inserir o bloco `Colour Sensor` no modo `Reflected Intensity Light` no começo do `Loop` . O nosso próximo passo é mostrar esse valor na tela do Brick; por isso, adicionaremos a variável `B` e o bloco do `Display` nessa respectiva

ordem. A princípio não se preocupe com a configuração, mais adiante veremos como realizar a configuração de cada um.

Precisamos configurar o sensor de cor para medir a intensidade refletida, fazemos isso da seguinte maneira:

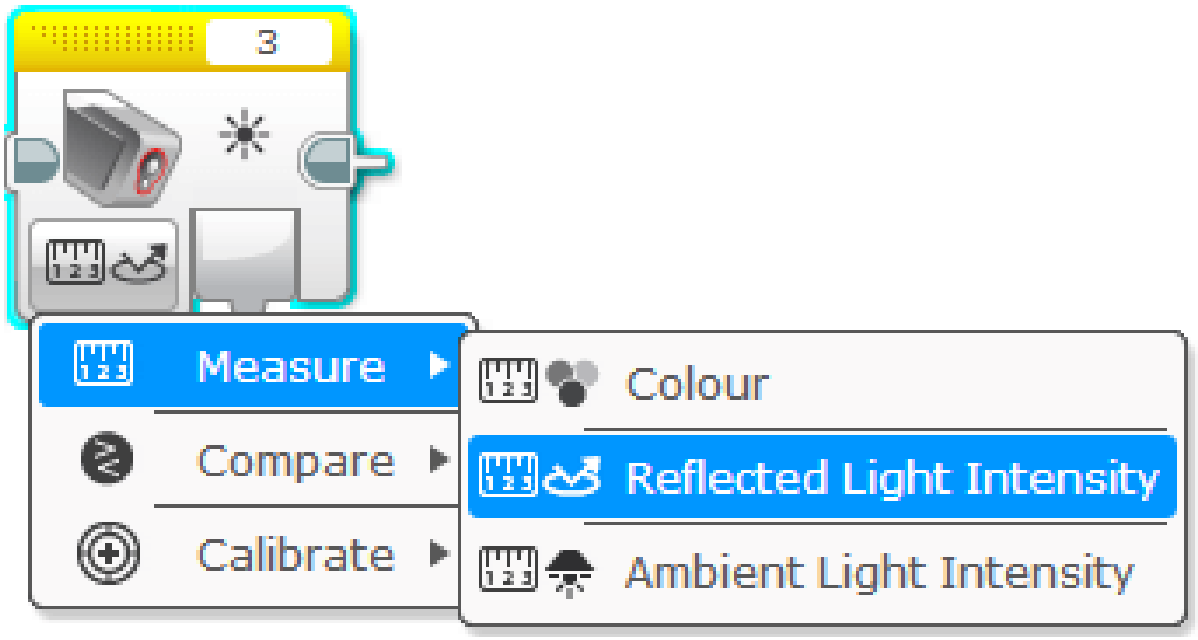



Figura 19.34: Solução da questão 3 - Configurando o bloco Colour Sensor

Após configurado o sensor, é necessário enviar o valor lido para a variável `B` . Para que esse valor seja escrito na variável `B` , ela deve ser configurada no modo `Write` .

O bloco `Display` receberá o valor lido pelo sensor (que foi armazenado na variável `B` ) e exibirá o número na tela do Brick. Logo, para que isso possa ser realizado, devemos conectar a saída `Value` do bloco `Colour Sensor` à entrada `Text` do bloco `Display` .

**Observação**

Também poderíamos ter adicionado novamente a variável `B` e tê-la configurado no modo `Read` . E, então, enviado o valor da variável `B` à entrada `Text` do bloco `Display` .

É importante ficar atento à configuração do bloco `Display` . Para que o valor enviado pela variável seja exibido, é necessário selecionar a opção `Wired` do `Display` . Dessa forma, o conteúdo enviado pelo fio de dados ( `Data wire` ) será exibido na tela.

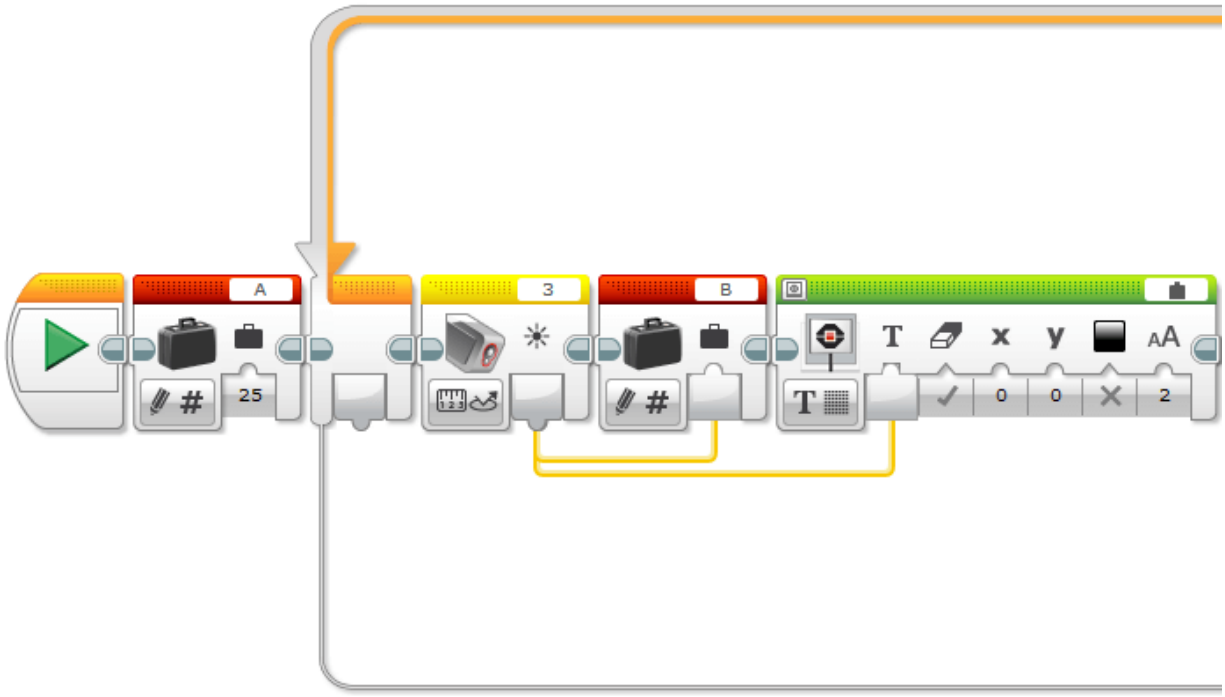



Figura 19.35: Solução da questão 3 - Adicionando o Loop

Agora começa a parte matemática do programa. Para que o nosso robô “saiba” qual motor acelerar e qual desacelerar (para que ele possa realizar o constante movimento de zigue-zague, e dessa forma, manter-se na trajetória) é necessário realizar uma comparação. As duas variáveis, `A` e `B` , são comparadas utilizando a mesma lógica do primeiro programa, o seguidor de linha.

Aqui está a configuração dos blocos.

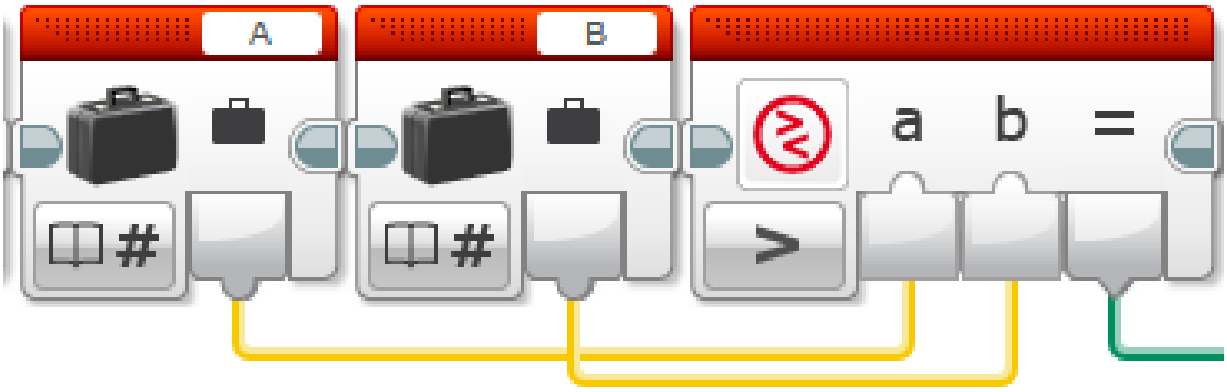



Figura 19.36: Solução da questão 3 - Parte matemática do programa

Adicionamos as variáveis `A` e `B` no modo `Read` , pois o bloco `Compare` lerá os valores das variáveis para poder realizar a comparação. No bloco `Compare` selecionamos a opção `Greater Than` (maior que), em seguida, conectamos o fio da variável `A` na entrada `a` do bloco `Compare` , e o fio da variável `B` na entrada `b` do bloco. O resultado dessa comparação é enviado para o `Switch - Logic` . Caso `A` seja maior que `B` , isso resultará em resultado lógico verdadeiro, então a linha de cima do `Switch` (que corresponde ao valor lógico verdadeiro, conforme foi explicado) é executada; caso contrário, a linha inferior é executada.

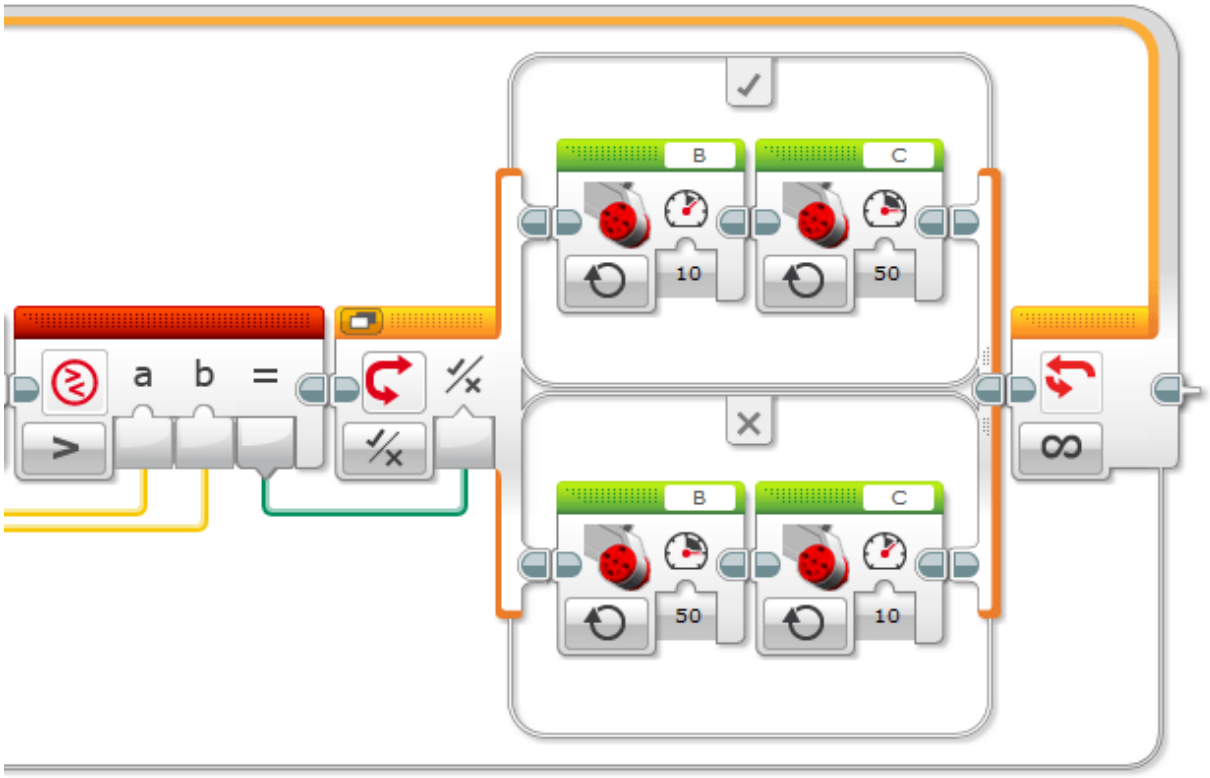



Figura 19.37: Solução da questão 3 - Comparação e Switch Logic

Acabamos de resolver o último exercício, porém, ainda faremos mais um programa. Estudaremos a seguir um bloco importantíssimo, o `Logic Operations` , situado na aba `Data Operations` . Na seção a seguir, veremos um pouco mais do bloco `Logic Operations` . Assim, você poderá conhecer todos os seus recursos e entender melhor cada um deles.

## 19.7 O bloco Logic Operations

Figura 19.38: Ícone do bloco Logic Operations

Este bloco é usado para operações lógicas, quando se necessita de um valor verdadeiro ou falso. Ou seja, o `Result` dele ou é `True` , ou `False` . O bloco possui as entradas `a` e `b` , exceto para o modo `Not` , que só tem a entrada `a` .

Os modos de operação são:

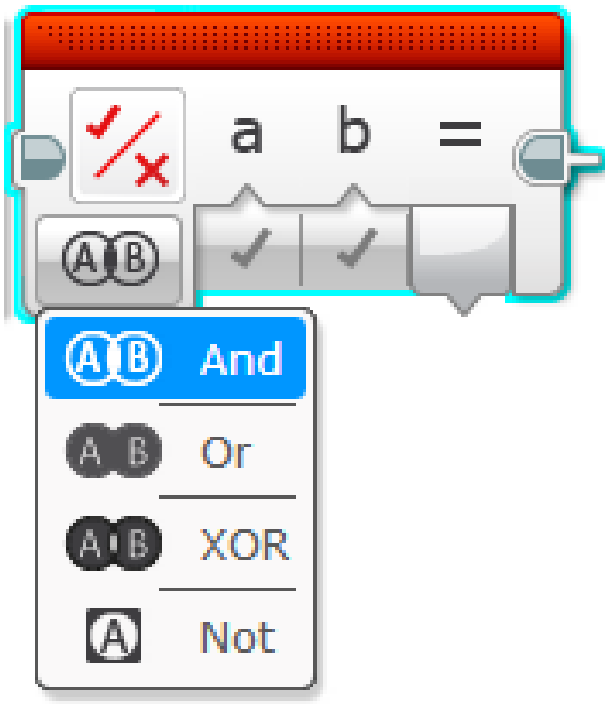



Figura 19.39: Modos de operação do bloco Logic Operations

- `And` : o resultado só é verdadeiro caso as entradas `a` e `b` sejam ambas verdadeiras. Caso contrário, o resultado é falso.

- `Or` : o resultado é verdadeiro caso uma das duas entradas ou ambas sejam verdadeiras. Quando as duas são falsas, o resultado é falso.
- `XOR` : resulta em um valor verdadeiro apenas se uma das duas entradas for verdadeira. Para as duas entradas verdadeiras, se obtém um resultado lógico falso. Para as duas entradas falsas também se obtém um resultado lógico falso.
- `Not` : (este possui apenas a entrada `a` ) O resultado é o oposto ao valor de entrada. Se `a` é verdadeiro o resultado é um valor lógico falso. Se `a` é falso, o resultado é um valor lógico verdadeiro.

Agora que você já conhece o funcionamento do bloco `Logic Operations` , criaremos um programa utilizando este bloco para que você possa ver seu funcionamento na prática.

## 19.8 Programa extra - Seguidor de Linha com mensagem no Display

Aqui nesta seção será proposta uma abordagem diferente. A maior parte dos conceitos utilizados para a criação desse programa já foi trabalhada por nós. Dessa forma, vamos fazer uma interpretação do código, porém, de maneira bem didática. Assim, você pode testar suas habilidades programando e interpretando o código.

### Objetivo do programa

Este programa que criaremos a seguir fará com que o robô mude de sentido a cada 8 segundos e, após a 3ª modificação de direção o número de mudanças será exibido no Display (na tela do Brick), juntamente com mensagem abreviada, indicando que o botão esquerdo deve ser pressionado para desligar o robô, enquanto o botão direito dará sequência às operações.

## Criando o programa

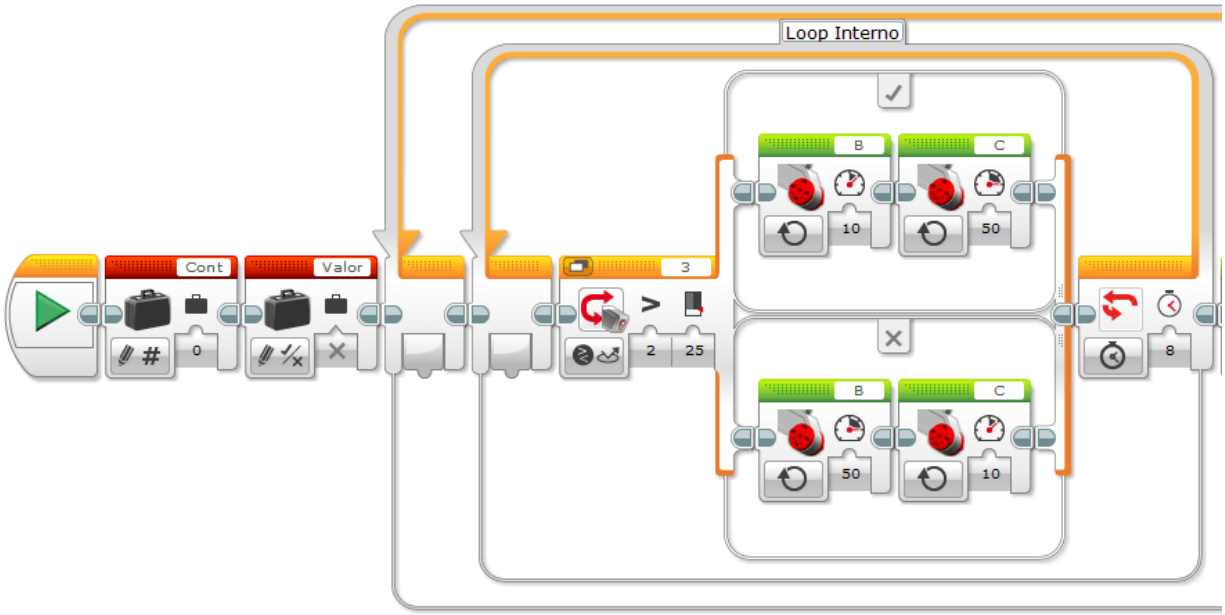



Figura 19.40: Seguidor Display

Após o bloco `Start` , temos a variável `Cont` , responsável por armazenar o número de detecções. Temos também a variável `Valor` no modo `Write – Logic` com um valor `False` escrito. A variável `Valor` é responsável por armazenar dados sobre o pressionamento do botão, esquerdo ou direito; mais à frente vamos entender melhor seu funcionamento. Seguindo, temos um `Loop` (externo), dentro do qual há um outro `Loop` com um `Switch` configurado da seguinte maneira: `Colour Sensor – Compare – Reflected Light Intensity` . Dentro desse `Switch` , configuramos o movimento de zigue-zague para que o robô siga a linha escura. O `Loop` interno foi configurado no modo `Time Indicator` e ajustado para 8 segundos, conforme o enunciado.

Quando os 8 segundos acabam, o `Loop` é interrompido. Os próximos dois blocos são o `Large Motor` , eles estão no modo `Off` , ou seja, esses blocos desligarão o motor. Logo em seguida está o `Move Steering` , que, conforme vimos, é responsável pelo giro do robô.

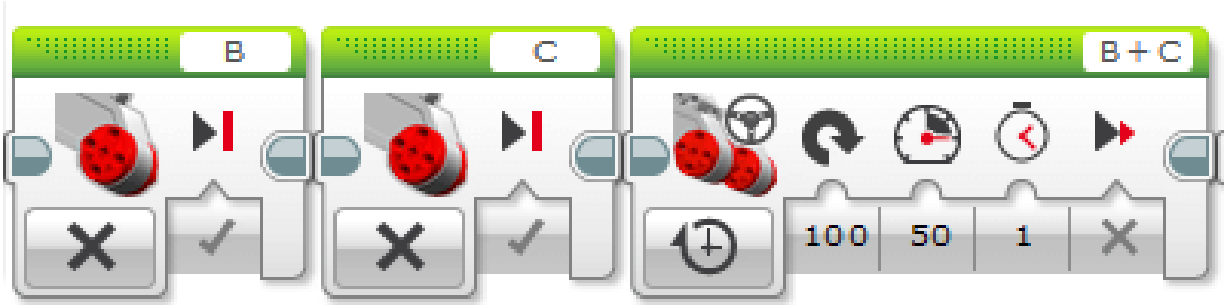



Figura 19.41: Seguidor Display - Blocos motores

Depois do bloco `Move Steering` , temos a variável `Cont` , que foi configurada no modo `Read - Numeric` . Assim, o valor é lido e, em seguida, enviado ao bloco `Math` , que devemos configurar no modo `Add` .

O valor lido deve ser enviado a entrada `a` (por meio do `Data wires` ), na entrada `b` devemos escrever o número `1` (pois a cada rodada o programa incrementa uma unidade à variável) que será adicionado ao valor da variável. Na saída `Result` do bloco `Math` está o resultado da operação, que é o que devemos enviar para a variável `Cont` , configurada no modo `Write - Numeric` . Dessa forma, a variável `Cont` vai ler o valor recebido.

Para que a comparação seja realizada, é necessário que o valor da variável `Cont` seja lido e enviado ao bloco `Compare` . Na entrada `b` do bloco vamos escrever o valor `3` (esse número é o número que servirá de parâmetro para a comparação), conforme o enunciado. O resultado da comparação deve ser enviado para o `Switch` , que antes devemos configurar no modo `Logic` . Esse `Switch` é responsável por exibir no Display o número de mudanças e a mensagem indicando que o botão esquerdo desliga o programa e o direito continua.

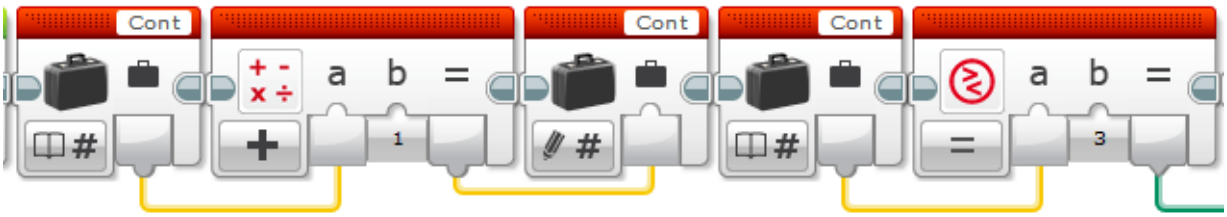



Figura 19.42: Seguidor Display - Variáveis

Dentro do `Switch` , na linha superior, que corresponde ao valor `True` (lembrando que, caso o valor seja `False` nada acontece, por isso não é necessário programar nenhuma sequência de blocos nessa linha), está o bloco `Move Tank` no modo `Off` . Logo em seguida, a variável `Cont` no modo `Read – Numeric` envia o seu valor para a entrada `Text` do bloco `Display` , com isso o número de contagens é exibido na tela do Brick. O bloco `Wait` no modo `Time Indicator` é responsável por fazer com que a mensagem seja exibida na tela por 2 segundos.

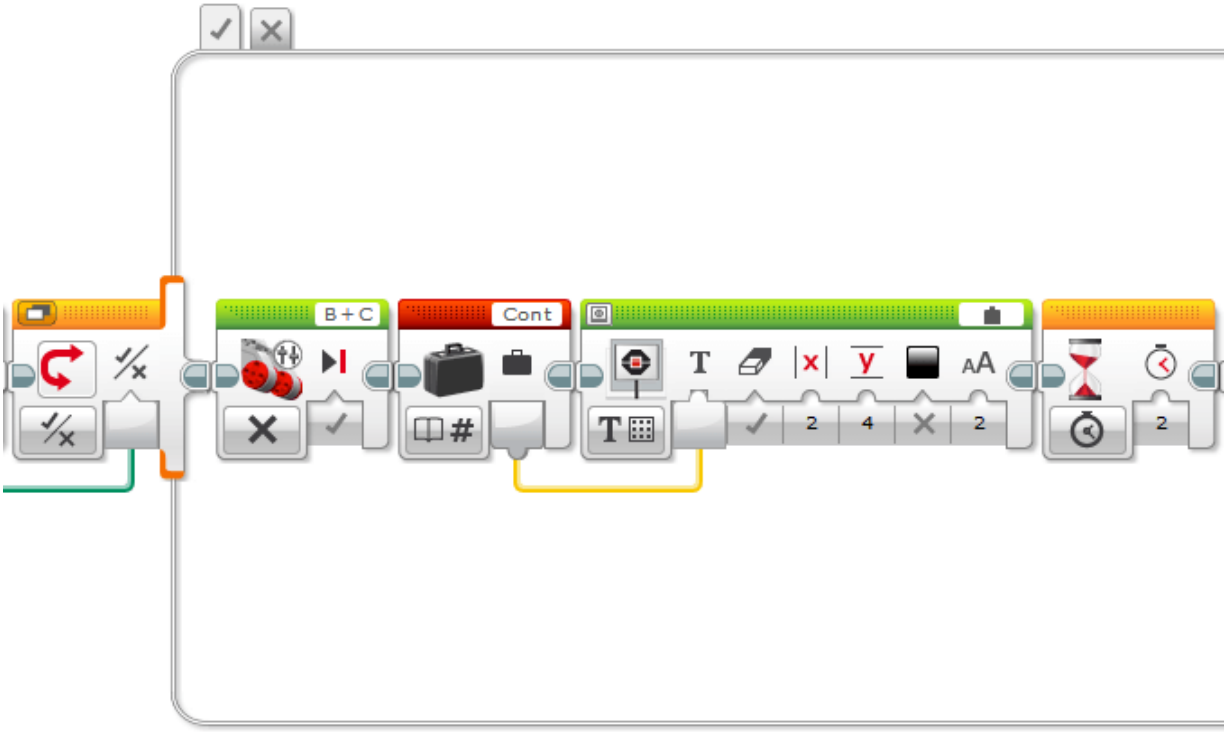



Figura 19.43: Seguidor Display - Exibindo a contagem no Display

Após o bloco `Wait` , a seguinte mensagem é exibida no Display: "Right - ON /Left – OFF". Logo após o bloco `Display` está o bloco `Wait` , que tem a mesma função que o bloco `Wait` anterior.

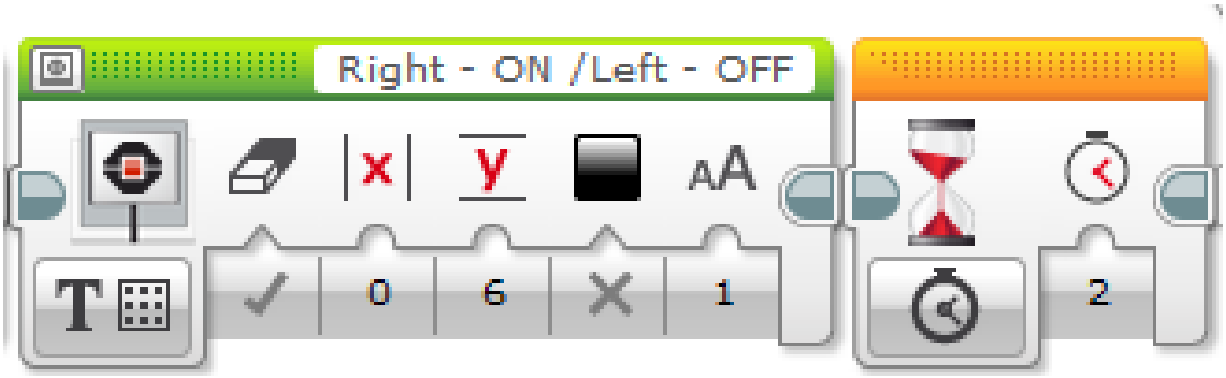



Figura 19.44: Seguidor Display - Exibindo a mensagem no Display

No `Loop - Logic` é feita a detecção dos botões. Para isso, utilizamos dois blocos sensores `Brick Buttons` , ambos configurados no modo `Compare -Brick Buttons` . Um sensor configurado para o botão esquerdo e outro para o botão direito. O estado ( `State` ) selecionado para os dois é o 1 ( `Pressed` ). O resultado ( `Result` ) deve ser enviado as entradas `a` e `b` do bloco `Logic Operations` , que foi configurado no modo `Or` (caso um, ou outro, seja pressionado é obtido então um valor lógico verdadeiro). O `Result` do `Logic Operations` deve seguir para o `Until True` do `Loop` . Ou seja, após um botão pressionado esse `Loop` é encerrado.

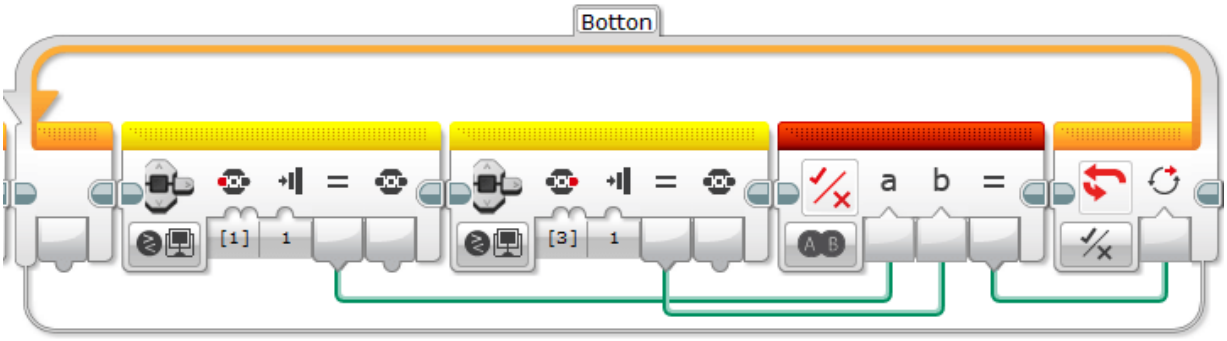



Figura 19.45: Seguidor Display

Quando o programa sai (ou seja, ele executou todas as tarefas desejadas até ali) do `Loop` lógico, ele entra no `Switch` . Esse `Switch` foi configurado no modo `Brick Buttons - Compare - Brick Buttons` .

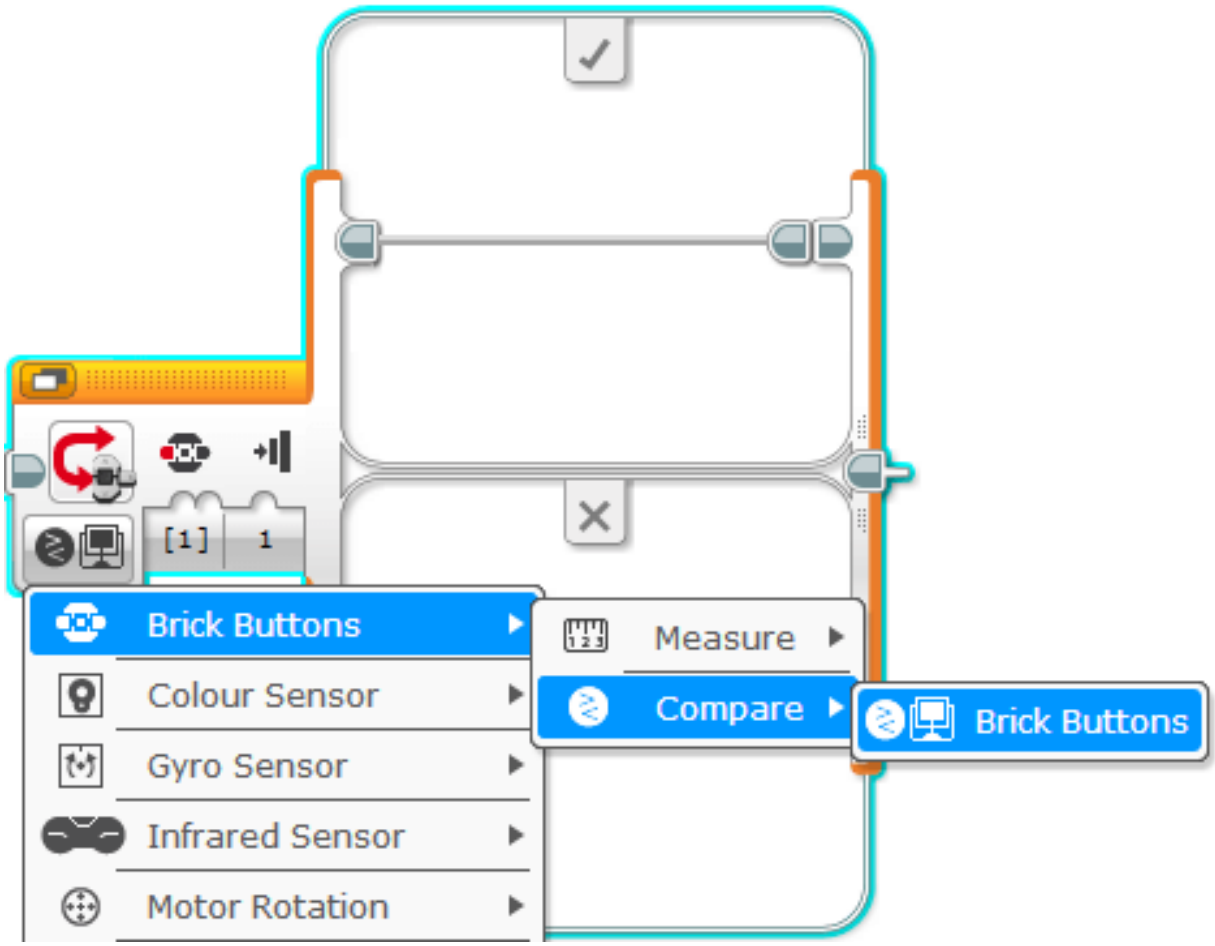



Figura 19.46: Seguidor Display - Compare

O botão esquerdo deve ser escolhido para a comparação ( `Left` ), ele deve ser configurado no estado `Pressed` . Caso o botão esquerdo seja pressionado, um valor lógico verdadeiro ( `True` ) será escrito na variável `Valor` (configurada no modo `Write - Logic` ); caso contrário, na aba inferior, correspondente ao resultado `False` do `Switch` , um valor `False` será escrito na variável. Após escrever o valor correspondente à ação na variável, o programa sai desse `Switch` , e sai também do `Switch` externo, o `Logic` .

Já no `Loop` externo, configuramos para o modo `Logic` . Em seguida, adicionamos a variável `Valor` no modo `Read - Logic` e enviamos seu valor para a entrada `Until True` do `Loop` . Dessa forma, quando a variável possuir um valor lógico `True` (botão esquerdo pressionado) o `Loop` será encerrado, e consequentemente o programa também.

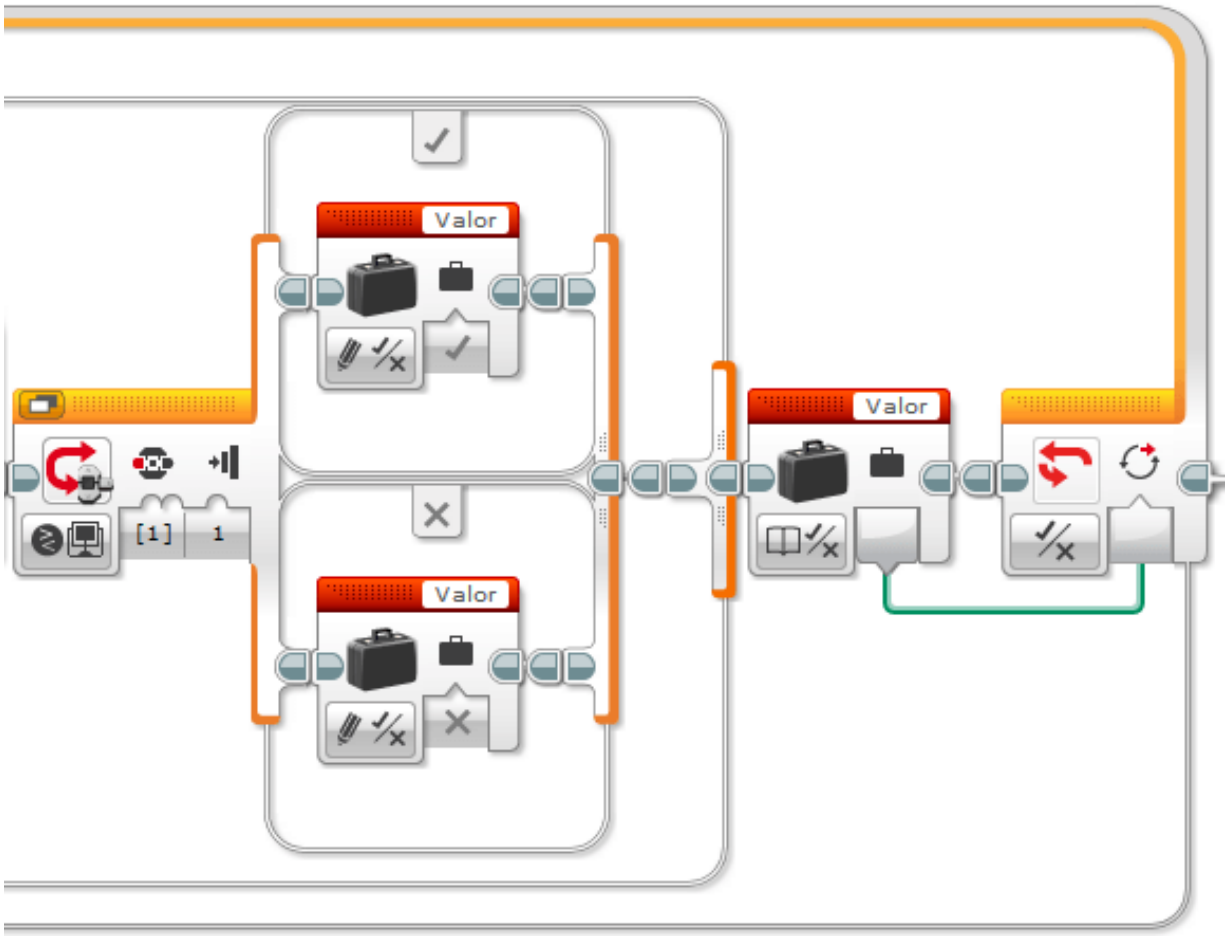



Figura 19.47: Seguidor Display

## 19.9 Movimentos

Com esta seção, você entenderá melhor o movimento do robô, e esclarecerá possíveis dúvidas, logo, será capaz de programar

movimentos melhores para seu robô, pois entenderá a lógica de funcionamento dos movimentos.

### Realizando um giro devagar à direita

Para fazermos com que o robô gire lentamente para a direita, realizando curvas mais abertas, ou seja, mais suaves, aplicamos velocidade máxima no motor da esquerda e velocidade menor no motor da direita. Desse modo, a roda direita, que corresponde ao lado direito, andará mais devagar, já o lado esquerdo se moverá mais rápido.

### Realizando um giro rápido à direita

Para fazermos com que o robô gire rapidamente para a direita, realizando curvas mais fechadas, aplicamos velocidade máxima no motor da esquerda e velocidade nula no motor da direita. Desse modo, a roda direita, que corresponde ao lado direito, não se movimentará, já o lado esquerdo se moverá acarretando em um movimento muito rápido para a direita.

### Realizando um giro em torno do próprio eixo

Aplicando potência máxima no motor da direita, porém, com valor negativo, e potência máxima positiva para o motor da esquerda, o lado direito do robô se move rapidamente para trás, e o lado esquerdo rapidamente para frente. Desse modo, o robô gira rapidamente em torno de seu próprio eixo.

### Pronto! E agora?

Esta foi nossa primeira experiência. Partimos de um programa que apenas fazia o robô seguir um traçado negro eternamente, e ao longo do capítulo fomos incrementando recursos novos. Você viu quais são os modos de funcionamento do sensor de cor e teve a oportunidade de testar na prática cada um deles.

Também trabalhamos com blocos novos, que serão importantíssimos para a construção dos programas que veremos adiante. Exploramos bastante a paleta *Data Operations*. Você aprendeu sobre o bloco `Math` , `Compare` e sobre o bloco `Logic Operations` . Além disso, reforçamos, ainda mais, o contato e experiência com as variáveis.

Ainda temos mais duas experiências pela frente. Nelas, teremos a oportunidade de conhecer ainda mais blocos e recursos. Mas, antes de prosseguir, você pode colocar em prática suas próprias ideias, criando modificações. Nos capítulos anteriores, diversas modificações foram propostas, você pode fazer o mesmo aqui. Tente criar variações próprias dos códigos aqui apresentados, explore as demais opções dos blocos que estudamos. Sinta-se livre e à vontade para modificar. Somente programando e testando você vai adquirir experiência e reforçar sua aprendizagem.

# Capítulo 20
# Sumô

Nesta experiência, faremos um Robô Sumô, cujo objetivo é permanecer em um perímetro delimitado e empurrar um robô adversário. Este robô está presente em muitas competições de robótica. Mas, não se assuste, você vai ver que sua programação é bem simples.

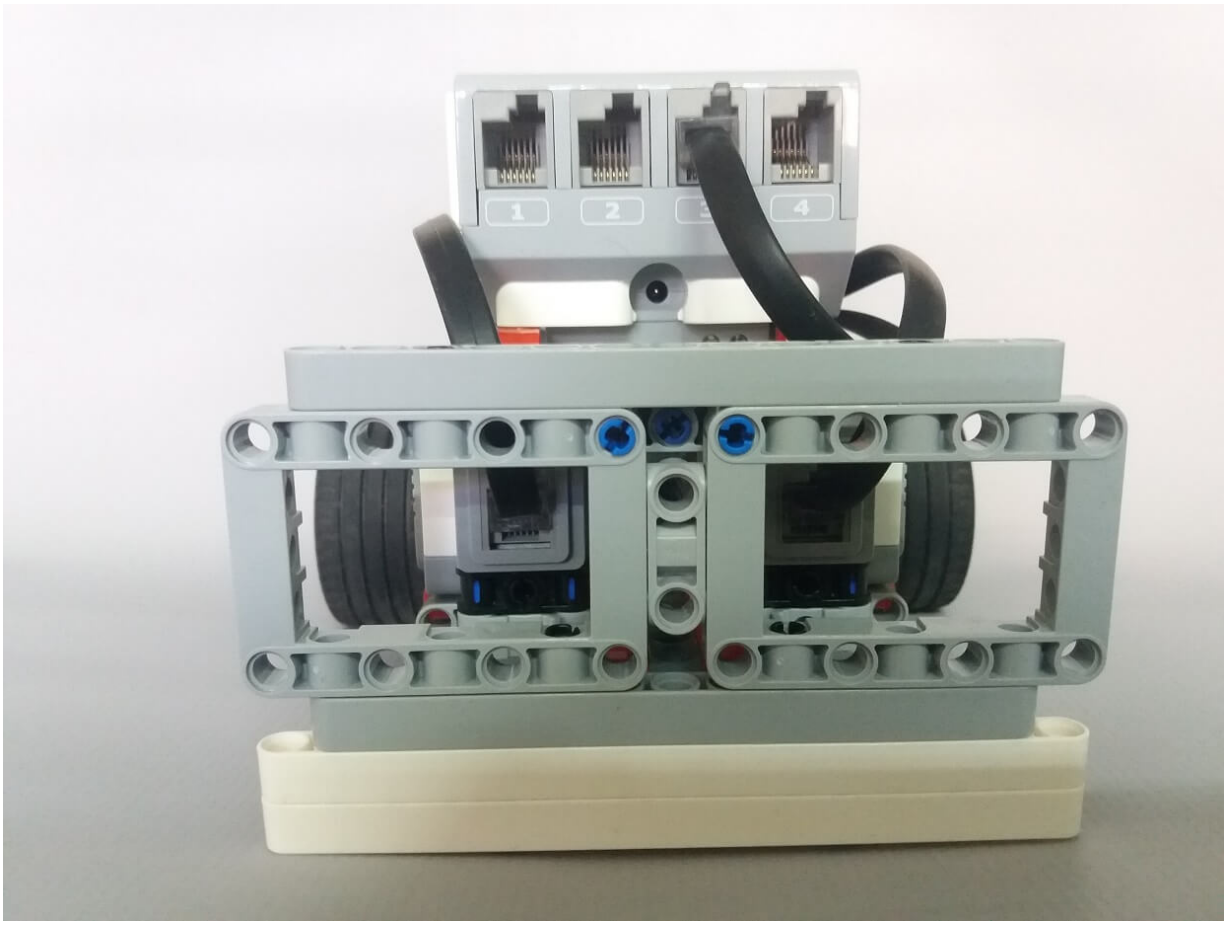



Figura 20.1: Robô Sumô

## Instruções de montagem

O robô da experiência anterior será utilizado para esta, porém, algumas alterações serão feitas nele. Conforme visto na imagem, uma estrutura semelhante a um para-choque foi acrescentada no robô. Na pasta `Instructions` desta experiência, você encontra as informações necessárias para montá-lo. O link de acesso a pasta é:

https://github.com/vitor000x/ev3_Codes/tree/master/Experiencias/2-Sumo/Instructions.

**Objetivo do programa**

Vamos imaginar um protótipo de Robô Sumô, colocado dentro de um circuito de luta demarcado por perímetro escuro. Para o circuito de luta podemos utilizar a pista que foi usada na experiência do robô seguidor de linha, pois é só posicionar o robô no meio da pista e, então, o traçado negro servirá para delimitar o perímetro. O Robô Sumô tem como objetivo empurrar um robô inimigo para fora de um perímetro delimitado e permanecer nesse perímetro.

No seguinte link você encontra fotos e informações referentes ao robô sumô do kit anterior, o NXT. No link você poderá ter maiores informações e poderá encontrar fotos que ilustram a experiência: http://nxtprograms.com/mini_sumo/index.html.

Vamos elaborar um código para controlá-lo, considerando que seu objetivo é movimentar-se em linha reta até perceber os limites do circuito, quando então o robô parará, recuará por um segundo, realizará meia volta e continuará seu deslocamento retilíneo, repetindo indefinidamente estas operações.

Para testar na prática a eficiência de seu robô você pode inserir alguns objetos dentro do perímetro. Dessa forma, quando o robô colidir com um objeto, ele empurrará o objeto para fora do perímetro.

Você verá que esse programa tem muitas semelhanças com o Robô Seguidor de Linha. Na verdade, parte da lógica é a mesma. Nosso objetivo é fazer com que o robô fique apenas no perímetro

demarcado. Para isso, utilizaremos o mesmo princípio do capítulo anterior.

## 20.1 O código

No Robô Seguidor de Linha, utilizávamos um `Switch` configurado no modo `Colour Sensor - Compare - Reflected Light Intensity` . Aqui também utilizaremos essa mesma configuração. Caso você não se recorde muito bem, este `Switch` tem por objetivo fazer a detecção da linha escura. Dessa forma faremos com que o nosso robô fique no perímetro delimitado pela linha negra do circuito. Após essa breve recapitulação e uma melhor compreensão da lógica do robô, vamos ao código.

Após o bloco `Start` , vamos inserir o bloco `Loop` para que o robô repita toda a sequência de blocos inseridos nele. Vamos configurar o `Loop` no modo `Unlimited` . O próximo passo é inserir o `Switch` , que foi discutido logo no começo da seção.

Agora é preciso configurar o `Switch` . No parâmetro `Compare Type` vamos escolher a comparação representada pelo número 2, a `Greater Than` (Maior que). Em `Threshold Value` definimos o valor limiar, nesse caso foi utilizado o valor `25` . Mas esse valor deve ser adequado à sua situação, dessa forma você garante o bom funcionamento do robô. No capítulo anterior já foi abordado como verificar a quantidade de luz refletida.

Conforme visto anteriormente, a linha negra da pista apresentava luminosidade inferior ao valor `10` , enquanto, a parte branca da pista reflete um valor superior a `80` .

Por esses motivos, utilizaremos o valor `25` no parâmetro `Threshold Value` ; pois o traçado negro que delimita o perímetro corresponde a um valor inferior a `25` , e a parte branca (interior do perímetro, fora do traçado negro), a um valor superior.

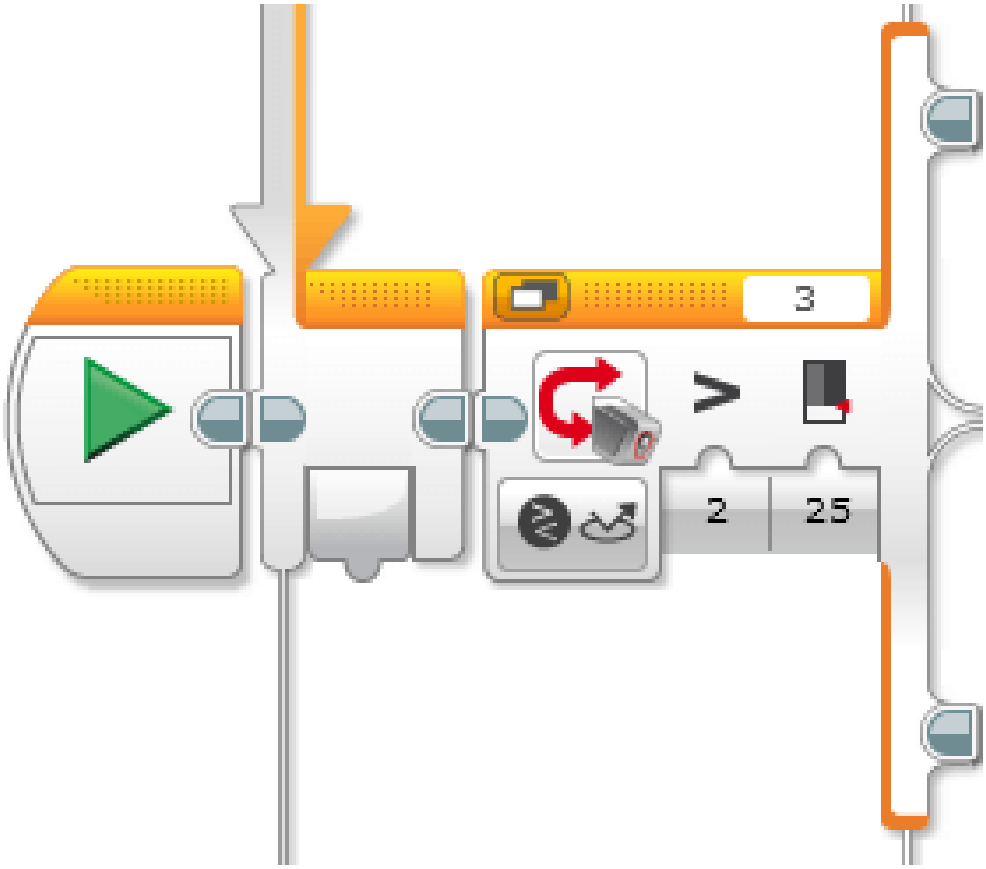



Figura 20.2: Robô Sumô: Switch configurado no modo Colour Sensor

Na linha superior do `Switch` ( `Case True` ) vamos inserir os blocos responsáveis pelas ações que serão executadas enquanto o robô se mantiver fora da linha negra, que delimita o perímetro; pois a parte branca do circuito apresenta uma luminosidade maior que `25` e, por isso, corresponde ao valor `True` do `Switch` .

Nesse `Case` vamos inserir o bloco responsável pelo movimento do robô. Para que o nosso robô ande, vamos utilizar o bloco motor `Move Tank` no modo `On` . Configurando o bloco no modo `On` ele executará suas ações até que a luminosidade seja menor que `25` . No parâmetro `Power` , vamos ajustar a potência dos motores para `50` , mas você pode mudar isso; apenas se lembre de que, caso o robô se mova muito rapidamente, as medições do sensor de cor podem ser prejudicadas.

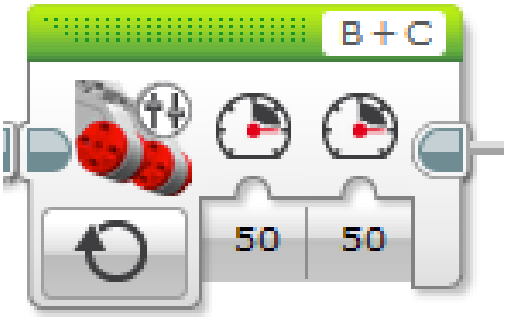



Figura 20.3: Robô Sumô: bloco Move Tank configurado com potência média

O `Case True` do `Switch` necessitou de apenas um bloco. Agora, vamos ao `Case False` (luminosidade menor que `25` ) do `Switch` , no qual vamos inserir os blocos responsáveis pela ré e pela volta do robô.

Quando o sensor do robô detectar uma luminosidade menor que `25` (ou seja, a linha negra) vamos primeiramente parar o robô, cessando o movimento de seus motores. Para isso, utilizaremos o bloco `Move Tank` configurado no modo `Off` . O segundo bloco é o `Wait` , com ele, vamos dizer ao nosso robô que ele deve esperar. Para isso, utilizaremos o modo `Time Indicator` e, no parâmetro `Seconds` , vamos inserir o valor `1` , assim ele espera por um segundo.

O bloco `Wait` , configurado para esperar por 1 segundo, é opcional. Não é necessário que o robô tenha que esperar para efetuar o movimento seguinte, que é o movimento de ré. Sendo assim, fica a seu critério adicionar o bloco `Wait` ou não. Sua única função é fazer com que o robô espere antes de executar a ação seguinte.

O terceiro bloco que vamos inserir é responsável pelo movimento de ré do robô. Mais uma vez utilizaremos o bloco `Move Tank` . Para que o robô recue por um determinado tempo utilizaremos o modo `On for Seconds` . Quando estudamos os blocos motores, vimos que para que o movimento seja executado em sentido contrário (para trás), basta que configuremos o parâmetro `Power` utilizando valores negativos (números antecedidos pelo sinal de menos: `-` ). No parâmetro `Seconds` , definimos a duração do movimento, que neste caso será 1 segundo.

Para finalizar o `Case False` do `Switch` , o nosso quarto e último bloco é novamente o `Move Steering` . Dessa vez, ele será responsável pelo movimento de giro do robô. Com esses movimentos o robô voltará à área branca, fora dos limites do perímetro. Para isso, vamos configurá-lo no modo `On for Rotations` , e no parâmetro `Steering` escolhemos o valor para rotação, que será `100` .

O parâmetro `Steering` (Rotação) é definido em uma escala de -100 a 100. Utilizando valores maiores que 0, o robô efetuará um movimento para a direita, já números negativos farão com que o robô efetue giro para a esquerda. Utilizando o valor 0, o robô simplesmente efetua um movimento em linha reta. Quanto mais próximo de 100, ou -100 (no caso de giros para a esquerda), mais fechado será o giro.

No parâmetro `Rotations` , definimos o número de vezes que o bloco executará o movimento e, neste caso, esse número é dado em rotações. Como queremos que o robô execute apenas uma volta, configuramos o parâmetro `Rotations` com o valor `1` . No parâmetro `Brake at End` , vamos selecionar a opção `Coast` , dessa forma o robô parará de forma suave. Mas, caso você queira, pode deixar a opção `Brake` selecionada.

O parâmetro `Break at End` pode ser configurado em `True` ( `Brake` ) ou `False` ( `Coast` ). No caso `True` , o motor para imediatamente e, se houver outro bloco depois dele, o bloco seguinte será executado apenas quando o primeiro terminar completamente seu movimento. No caso `False` , o motor para suavemente, e o bloco seguinte executa sua ação enquanto o motor ainda está parando.

Observe como ficou o `Case False` do `Switch` :

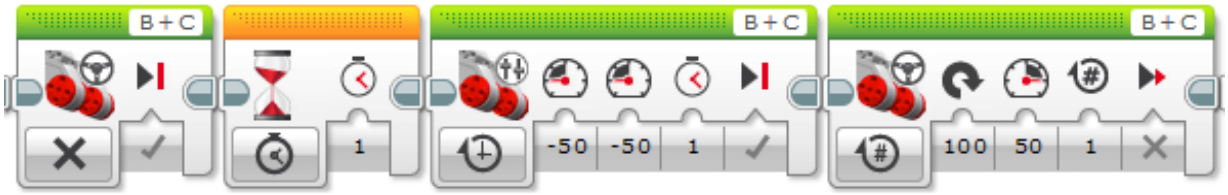



Figura 20.4: Robô Sumô: Case "False" do Switch

Agora veja como ficou o nosso programa completo:

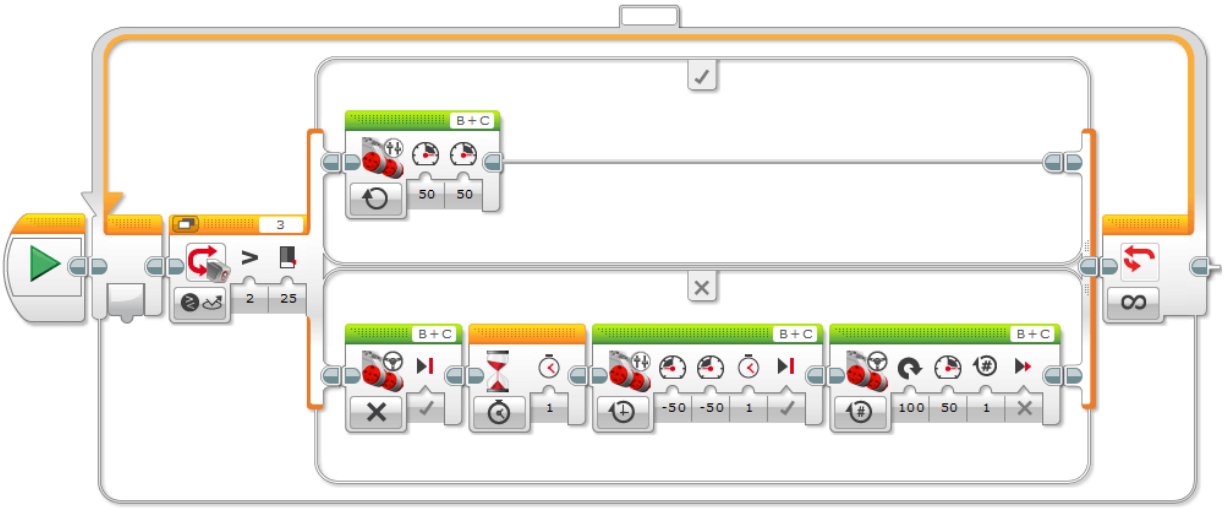



Figura 20.5: Robô Sumô: programa completo

Pronto, está finalizado o nosso programa. Agora é só testar, e caso precise, realizar os ajustes necessários.

## 20.2 Variação do Robô Sumô

No programa que faremos a seguir, realizaremos algumas alterações no programa do Robô Sumô utilizando variáveis e blocos de dados. Nesta variação do programa, o robô será programado para que, após realizar duas detecções da cor escura, ele seja encerrado. Agora vamos ao código!

Nosso primeiro passo será criar uma variável. Vamos criar uma variável chamada `Contador` , que será responsável por armazenar as detecções. Ela será do tipo numérica, logo, devemos configurá-la no modo `Write – Numeric` . Vamos adicioná-la após o bloco `Start` , e antes do `Loop` . Dessa forma, sempre que o programa for executado, a variável `Contador` iniciará com valor `0` .

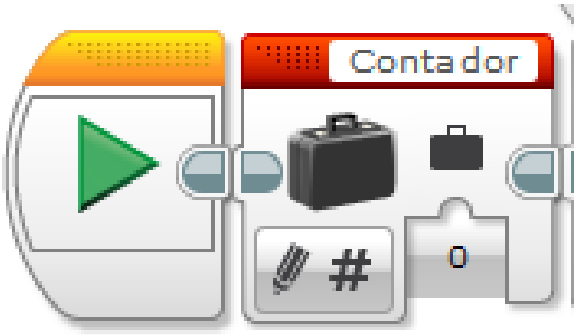



Figura 20.6: Robô Sumô: variável Contador

No `Case False` do `Switch` (linha inferior do `Switch` ), conforme já vimos, estão as ações executadas quando a linha escura é detectada. Nesse `Case` vamos inserir os blocos responsáveis pela contagem das detecções. Faremos isso utilizando a variável `Contador` e o bloco `Math` .

Caso você não se lembre como fazer isso, vamos recapitular.

Primeiro, devemos adicionar a variável `Contador` e configurá-la no modo `Read - Numeric` , pois o bloco `Math` lerá seu valor. Este bloco deve ser adicionado após a variável `Contador` e já apresenta configuração inicial no modo `Add` . Para fornecer o valor da variável ao bloco, ligamos a saída `Value` da variável à entrada `a` do bloco `Math` .

Com isso, o bloco `Math` lerá o valor da variável, mas agora é necessário fazer com que ele incremente o valor `1` a cada detecção. Para isso, vamos escrever na entrada `b` o valor `1` . Para que esse valor seja escrito na variável `Contador` precisamos adicioná-la novamente, desta vez, no modo `Write - Numeric` . O último passo é ligar a saída `Result` do bloco `Math` à entrada `Value` da variável `Contador` .

Pronto, agora, sempre que o `Case False` do `Switch` for executado, uma unidade será registrada na variável `Contador` .

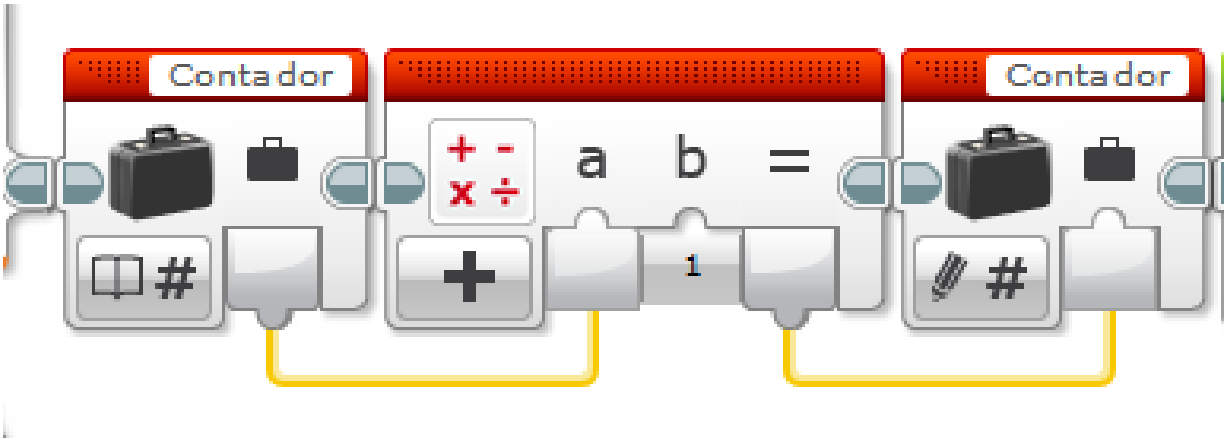



Figura 20.7: Robô Sumô: sequência de blocos responsáveis pelo incremento

Assim, depois de armazenar o valor na variável `Contador` , o robô executará as ações designadas para a detecção da cor escura, que são parar, esperar e dar a volta.

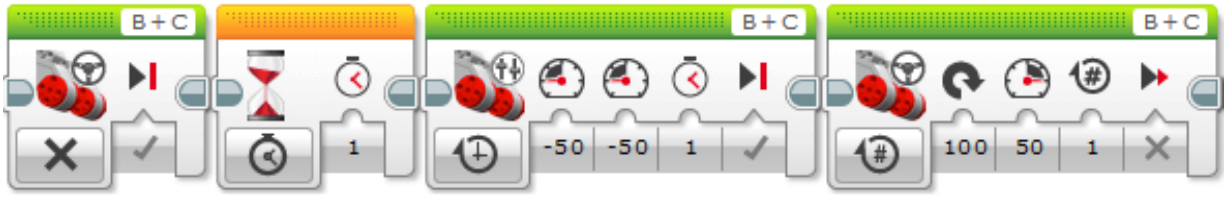



Figura 20.8: Robô Sumô: Case "True" do Switch

O próximo passo é configurar o `Loop` para que a ação do `Switch` seja executada até que sejam feitas duas detecções. Para isso, devemos configurar o `Loop` para o modo `Logic` . Dentro do `Loop` vamos adicionar a nossa variável `Contador` no modo leitura ( `Read - Numeric` ) e, em seguida, enviar o seu valor para o bloco `Compare` , que devemos configurar no modo `Equal To` . O parâmetro `a` recebe o valor da variável `Contador` , já a entrada `b` recebe o número que será o parâmetro de comparação, nesse caso o `2` .

O resultado do bloco `Compare` deve ser enviado então ao `Loop` .

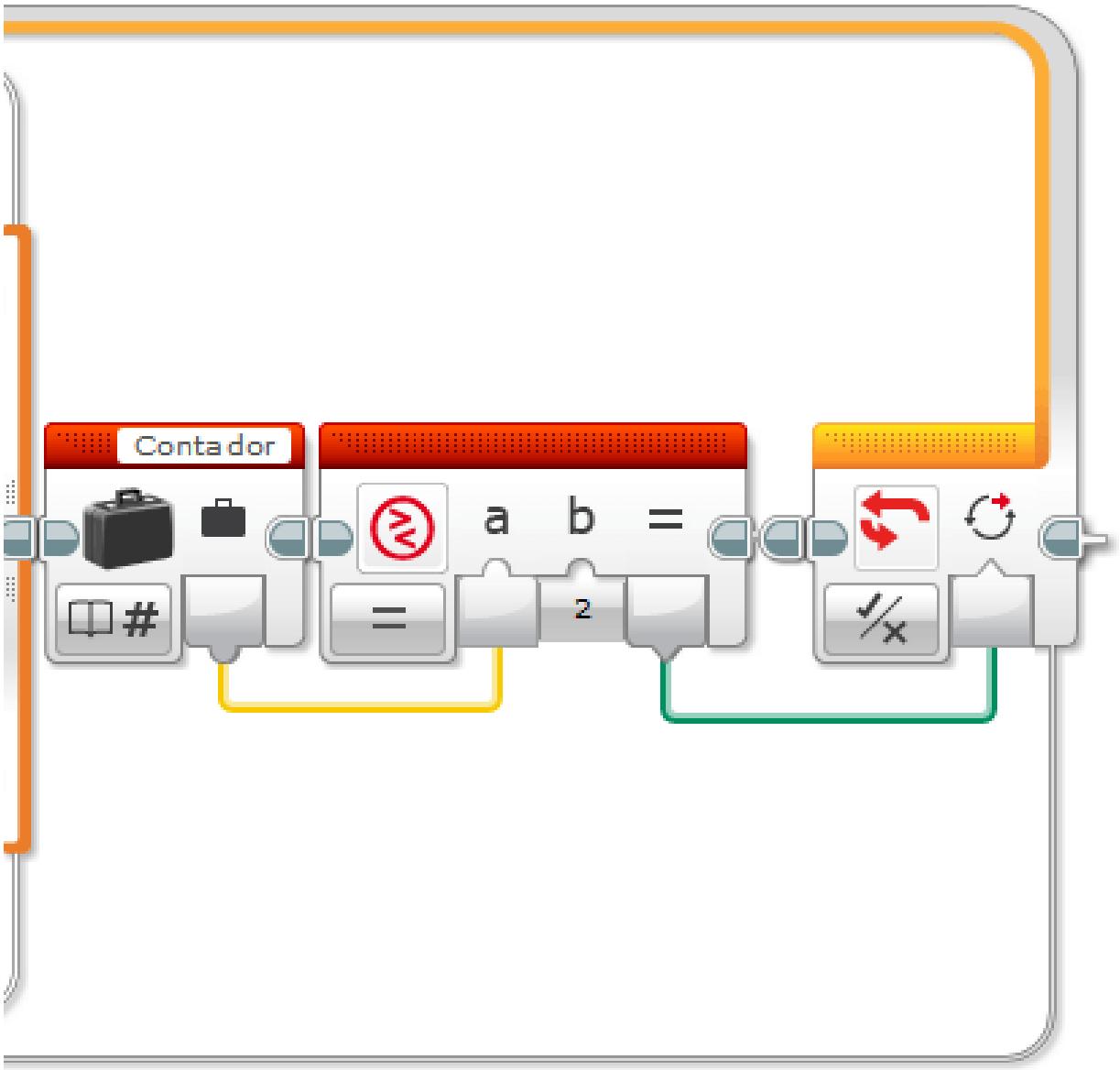



Figura 20.9: Robô Sumô: Blocos responsáveis pelo controle lógico do Loop

Aqui está o código:

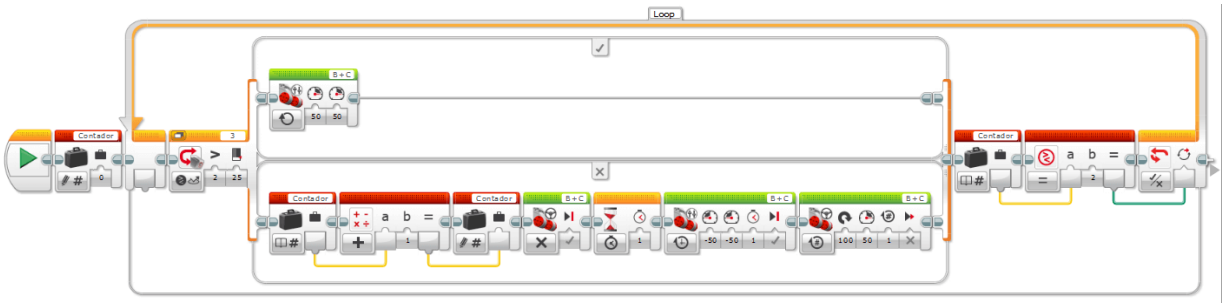


Figura 20.10: Variação do Robô Sumô: programa completo

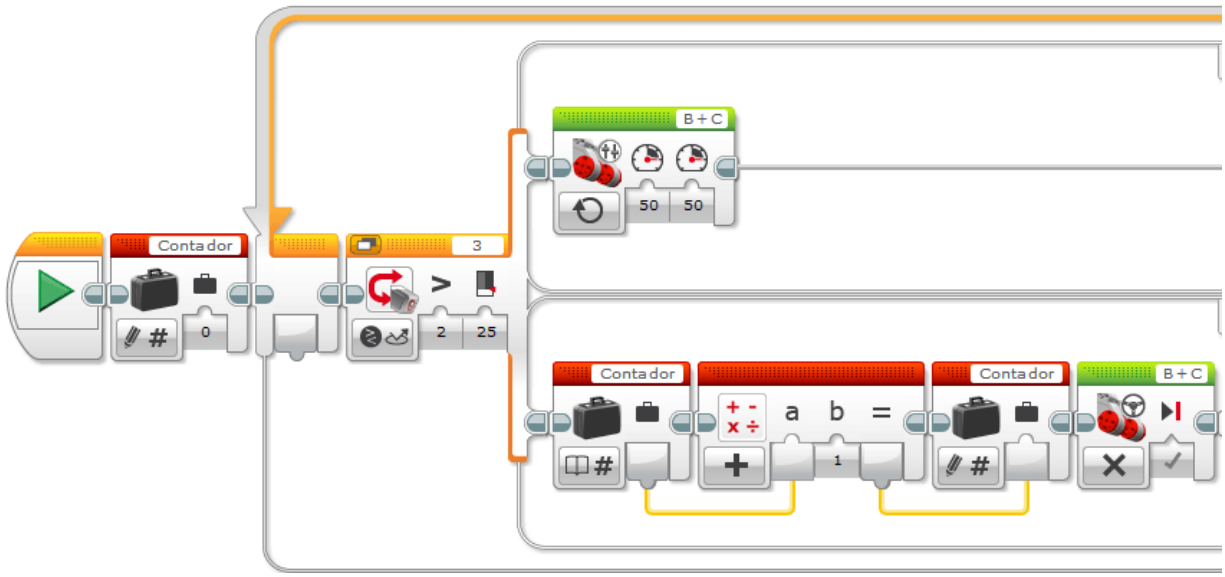



Figura 20.11: Variação do Robô Sumô - Detalhe 1

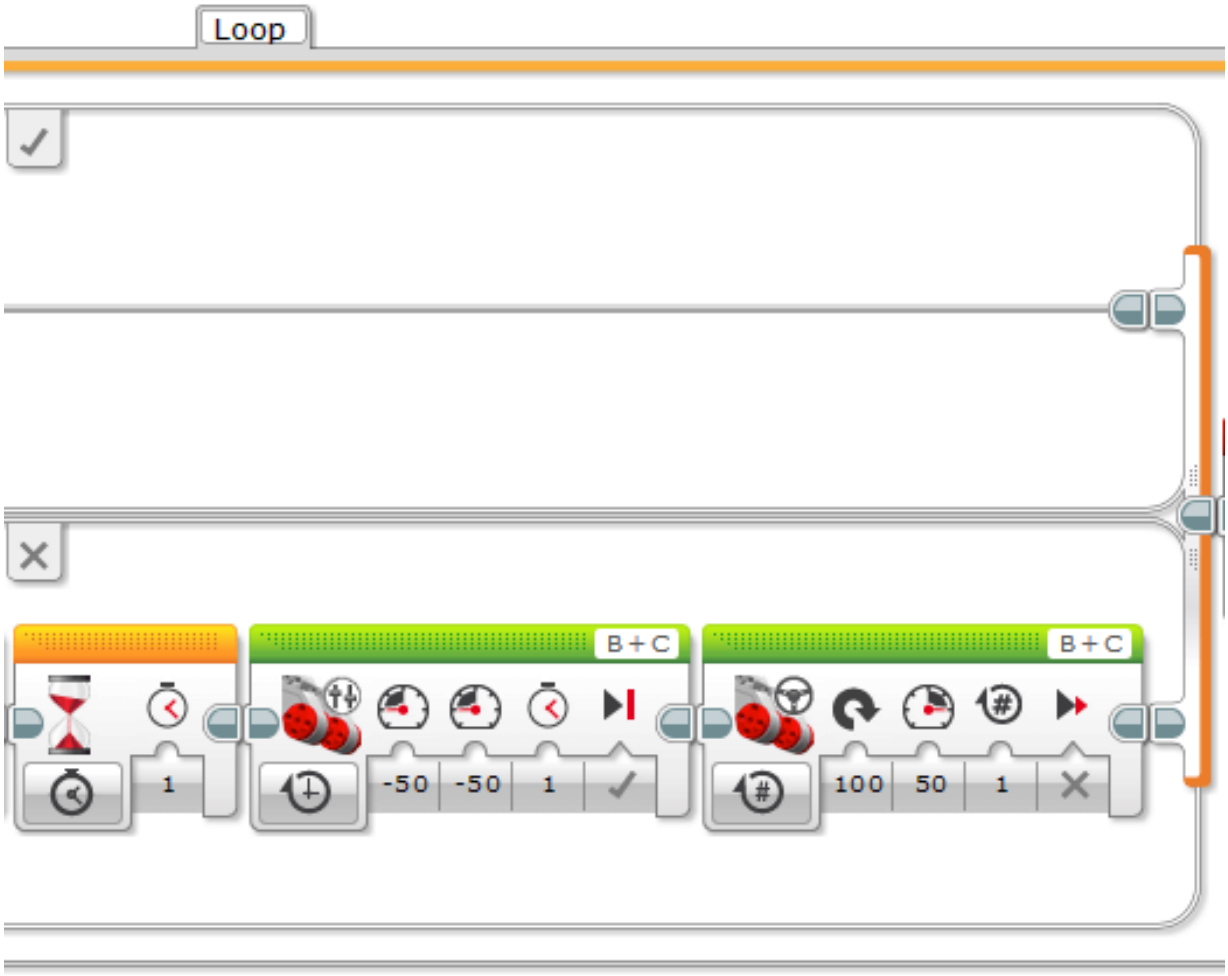



Figura 20.12: Variação do Robô Sumô - Detalhe 2

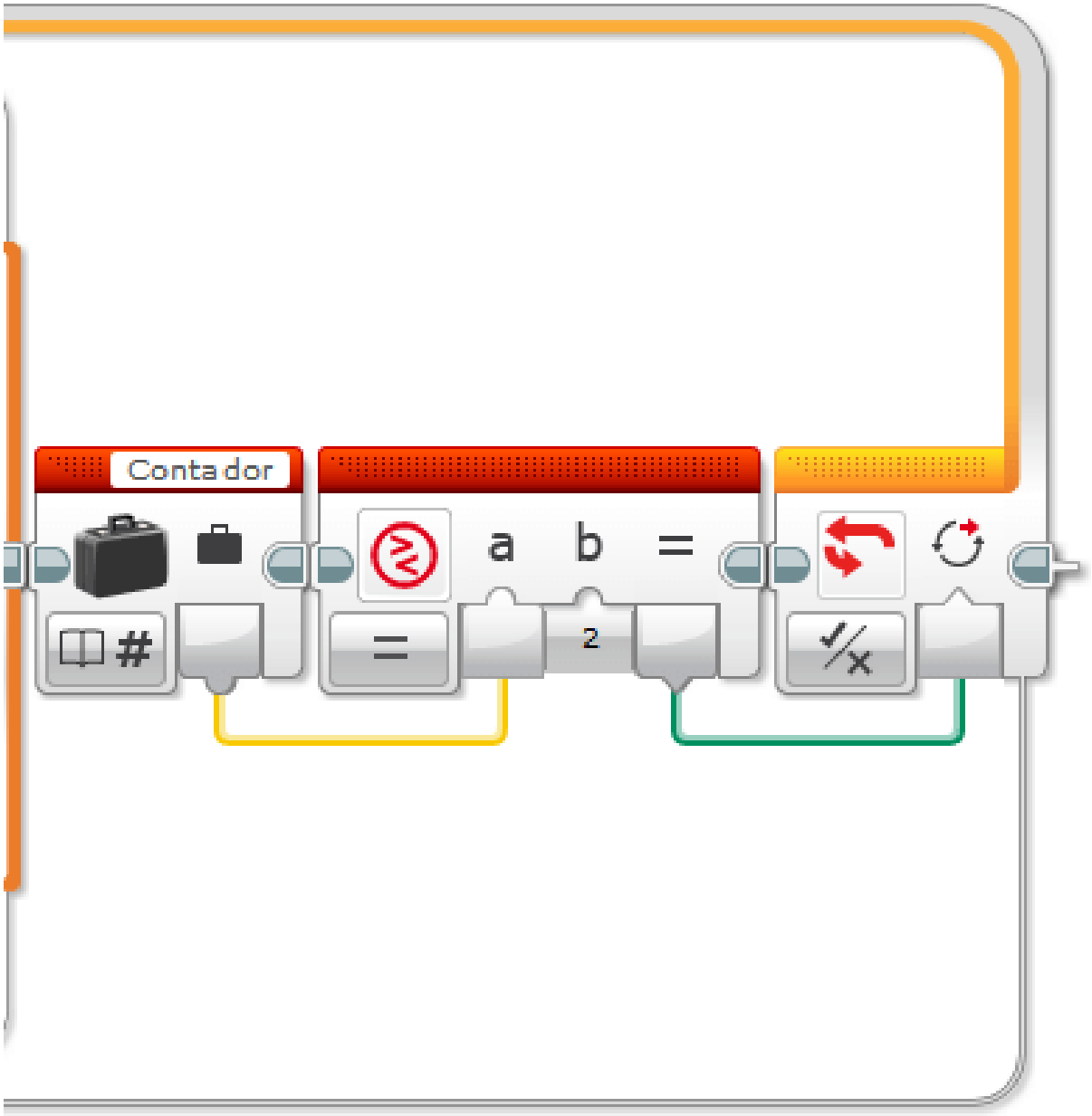



Figura 20.13: Variação do Robô Sumô - Detalhe 3

Em um dos exercícios a seguir você utilizará uma ferramenta muito importante. Trata-se da ferramenta `My Blocks Builder`, utilizada para criar blocos - nos capítulos anteriores já a estudamos um pouco. Mas, antes de ir para os exercícios, vamos estudar essa ferramenta mais a fundo e aprender como utilizá-la.

## 20.3 My Blocks

Já percebemos que, muitas vezes, na criação de um programa, um conjunto de comandos já escritos anteriormente são repetidos. Para poupar tempo e permitir uma melhor visualização do programa, pode-se resolver este problema de um modo bem simples e eficaz: criando um bloco próprio (que conterá toda uma sequência de blocos) que se encontrará na paleta `My Blocks` . Nesse bloco estão todos os blocos criados com a ferramenta `My Block Builder` .

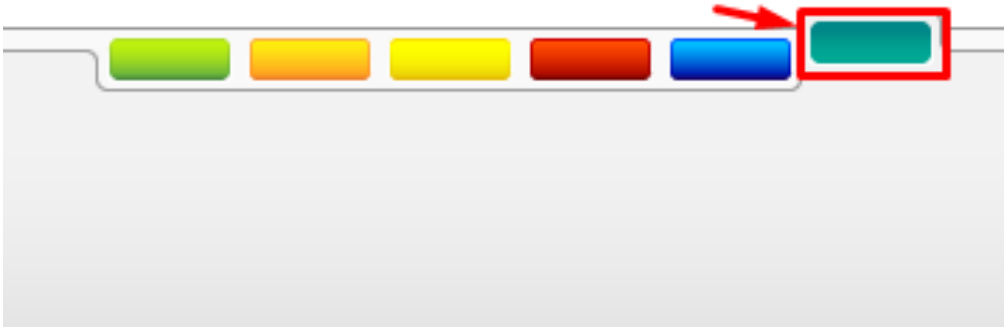


Figura 20.14: Paleta My Blocks - Note que a paleta está vazia, pois ainda não foi criado nenhum bloco.

O My Block Builder consiste em uma ferramenta na qual se pode agrupar um conjunto de comandos em bloco único, e utilizar esse bloco sempre que necessário.

### Criando um bloco próprio com o My Block Builder

Observe o programa a seguir.

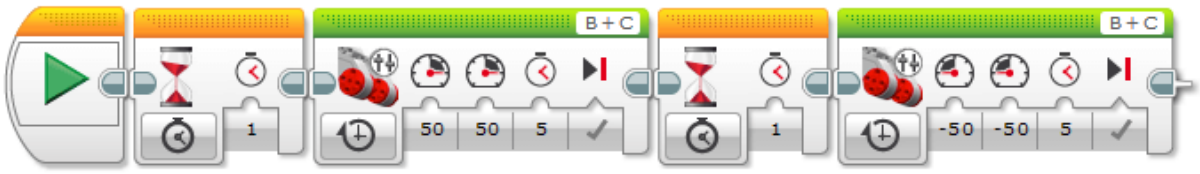



Figura 20.15: Exemplo de um programa

Este é um programa simples, que servirá para explicar o processo de criação de um bloco. Esse programa possui dois blocos `Wait` e dois blocos `Move Tank` . Ele faz com que o robô aguarde um segundo e execute um movimento para frente, em sentido horário. Logo após, o robô aguarda mais uma vez, por um segundo também. A seguir, executa o movimento em sentido contrário, ou seja, para trás (perceba que as potências estão com valor negativo).

Imagine que você precise ou queira utilizar essa sequência de blocos (a sequência a partir do bloco `Start` ) novamente. Seria necessário inserir todos aqueles blocos. Uma forma de reduzir o trabalho é transformando a sequência de blocos em um único só bloco.

Para criar um novo bloco, primeiro, devemos selecionar o conjunto de comandos que serão transformados em bloco único.

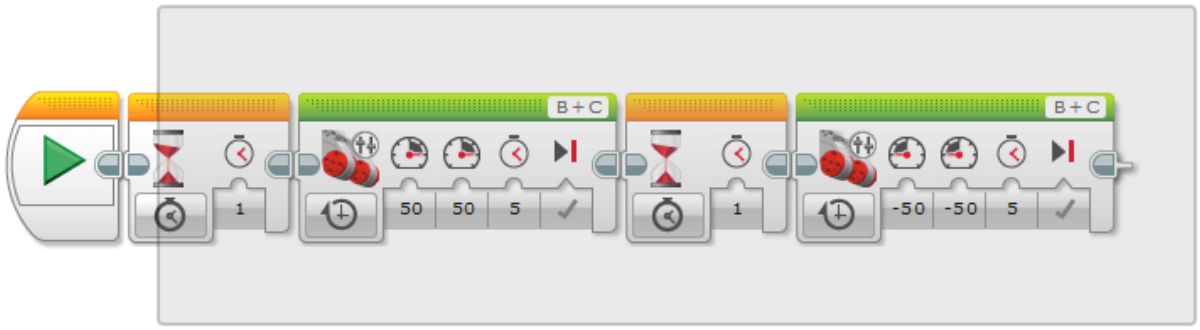



Figura 20.16: Selecionando os blocos

Quando os blocos são selecionados eles aparecem da seguinte maneira, conforme demonstra a figura a seguir.

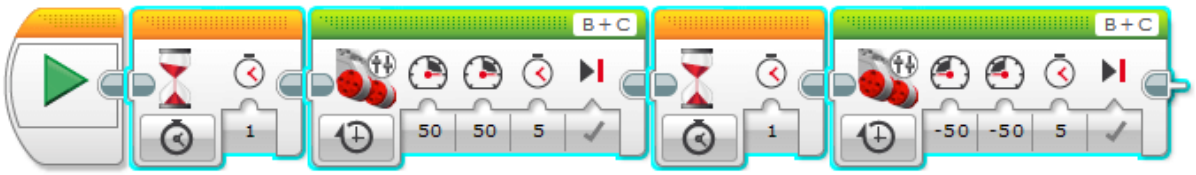



Figura 20.17: Blocos selecionados (estão com a borda azul em volta)

Feito isso, clica-se na aba *Tools* no canto superior direito do software. Em *Tools* deve-se clicar na opção `My Block Builder`. A seguinte tela aparecerá:

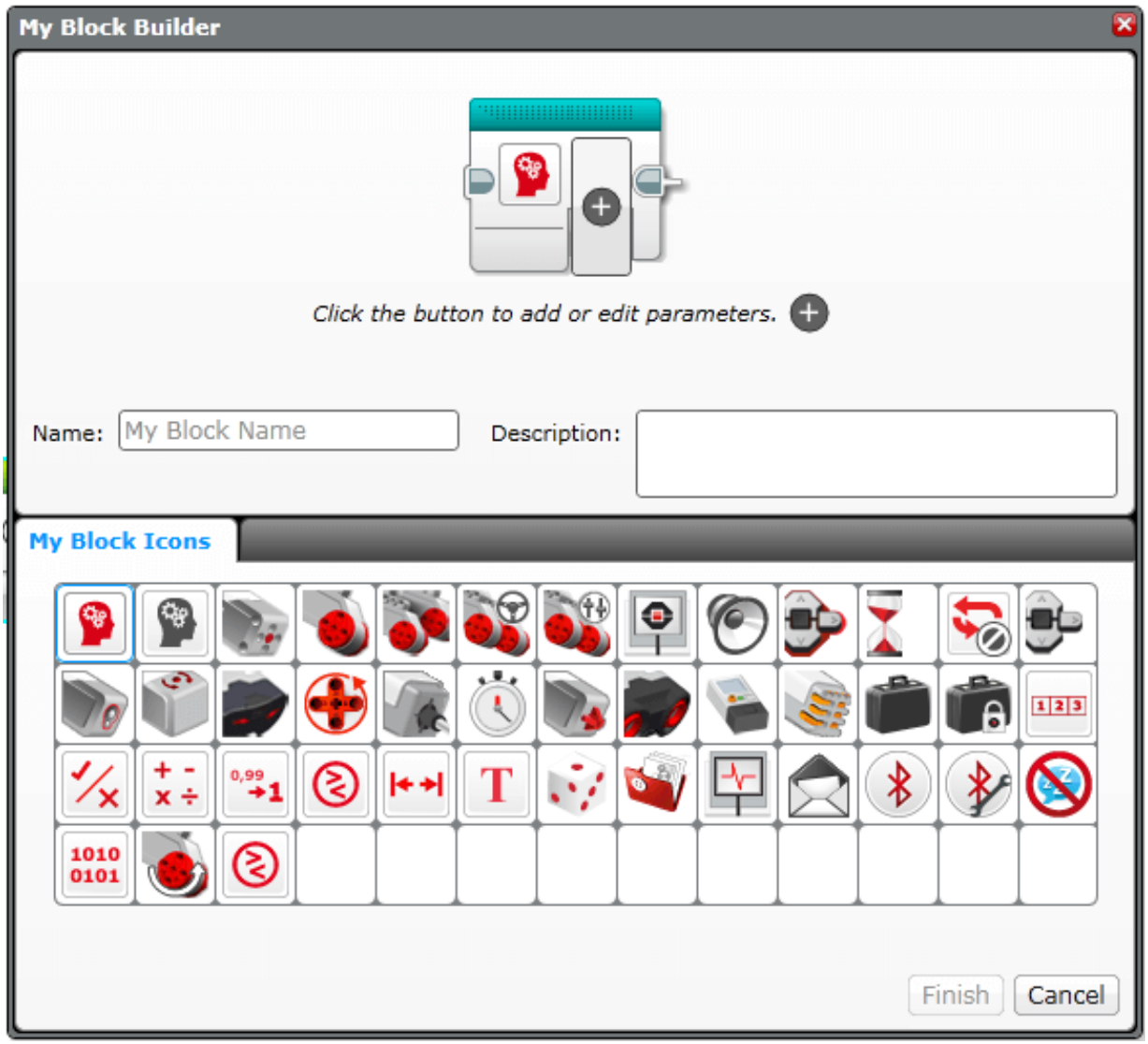



Figura 20.18: Ferramenta My Block Builder

Nessa tela pode-se atribuir nome ao bloco (na caixa `Name` ), adicionar uma descrição (em `Description` ) e associar um ícone ao bloco (no `My Block Icons` ).

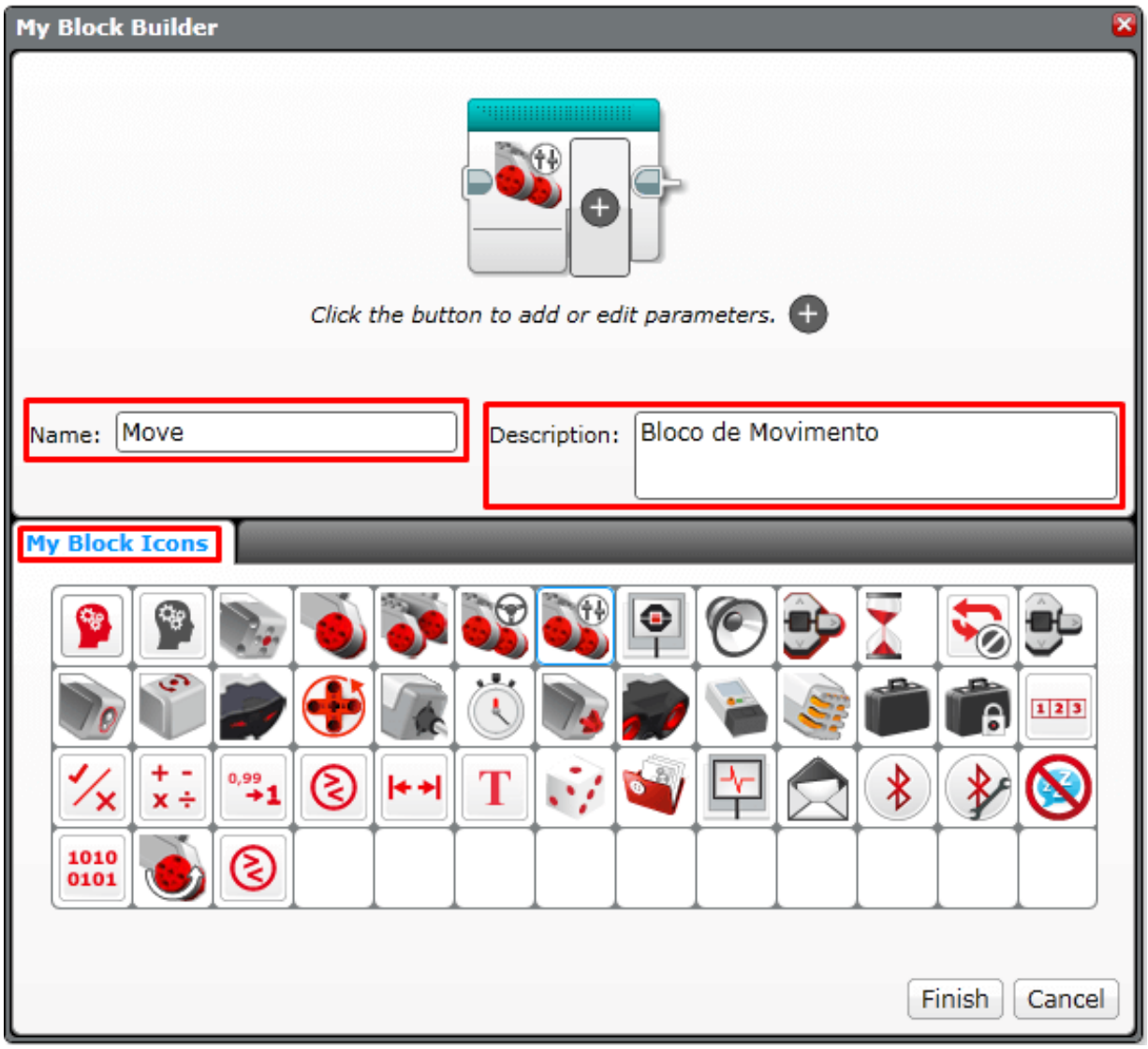



Figura 20.19: Configurando a ferramenta My Block Builder

Após escolher o nome, adicionar uma descrição e escolher um ícone, o bloco está criado. Observe o resultado a seguir.

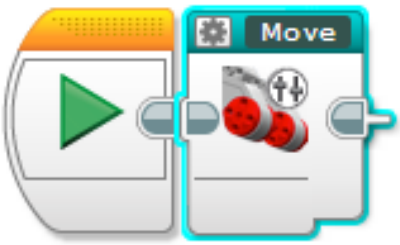



Figura 20.20: Bloco "Move" criado

## 20.4 Exercícios

1. Considere o protótipo de Robô Sumô que fizemos no início do capítulo. Realize as alterações necessárias ao programa para que o mesmo repita suas operações por duas vezes consecutivas, considerando a realização da contagem após o robô dar meia volta. Após contabilizar duas operações, o robô dará meia volta e parará, reiniciando suas atividades após 5 segundos, quando então repetirá o ciclo anterior por três vezes. Ao final das ações anteriores, o contador de repetições volta a ser dois, e as operações são novamente repetidas, até que o contador retorne ao valor três.

2. Repita o item anterior, utilizando o `My Blocks` na primeira contagem.

3. Repita o programa apresentado em *Varia  o do Rob  Sum* mostrando na tela do EV3 o número de detecções de cores escuras e gerando sons que informem a contagem.

## 20.5 Resoluções

**1.**

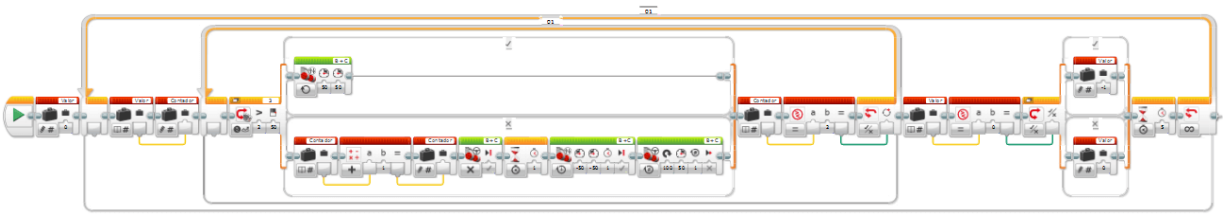


Figura 20.21: Robô Sumô: Exercício 1 - Programa completo

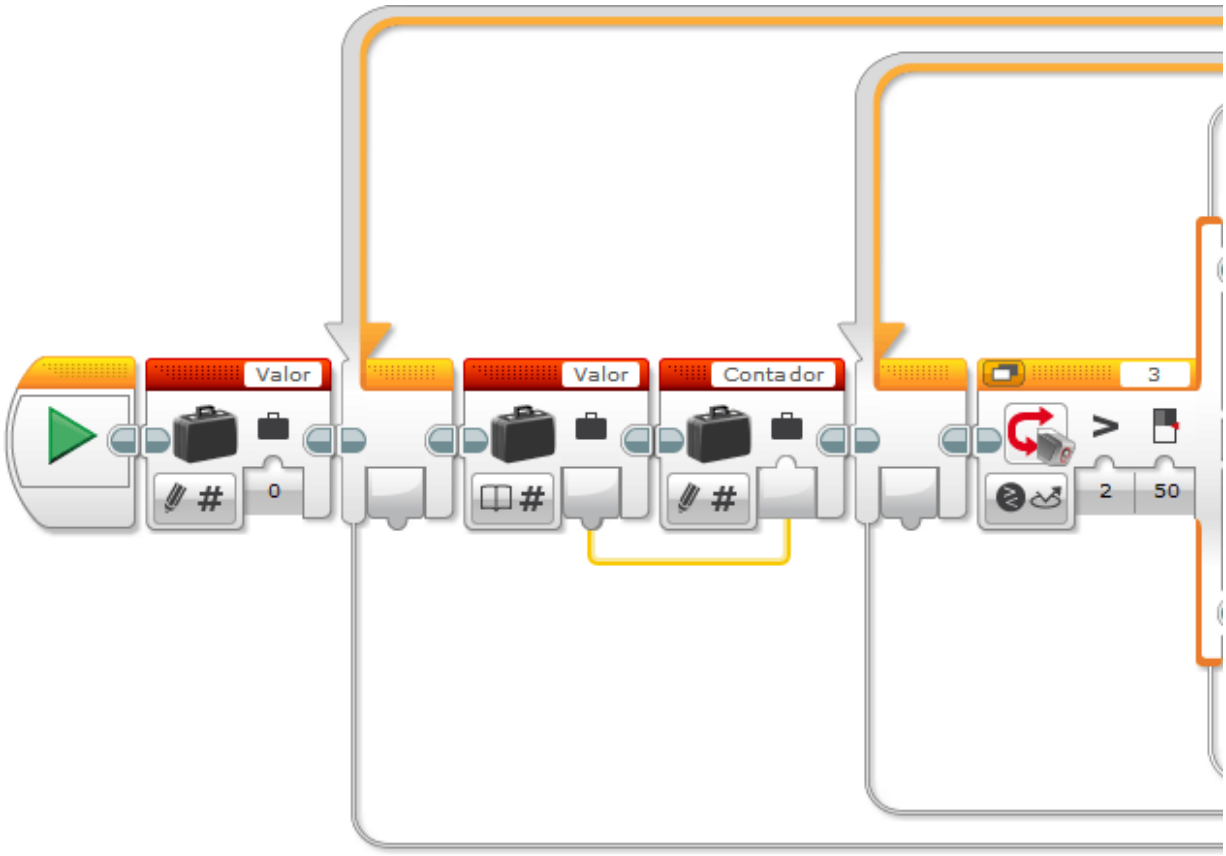



Figura 20.22: Robô Sumô: Exercício 1 - Detalhe 1

A próxima parte do código, o `Switch` , contém a mesma sequência de operações que já foi vista, nele está o movimento básico do robô, que é executado 2 vezes. Por isso, vamos à parte final do código.

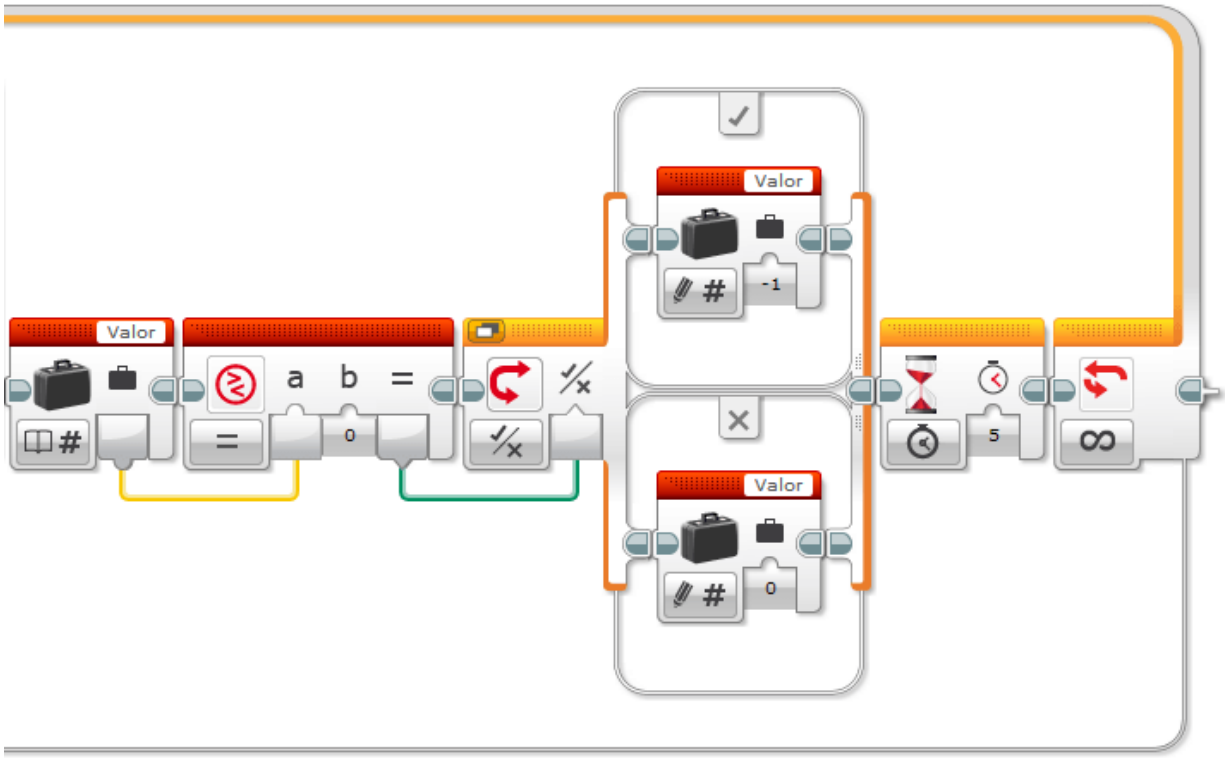



Figura 20.23: Robô Sumô: Exercício 1 - Detalhe 2

Para este exercício vamos utilizar como base o programa já feito. A partir dele faremos todas as modificações necessárias, para que se atenda aos requisitos do enunciado. Utilizaremos duas variáveis. A primeira variável, chamada de `Valor` , será responsável por repetir 3 vezes as ações após o robô executar duas detecções. Nesta variável, selecionamos o modo `Write - Numeric` , e escrevemos o valor `0` nela. Ela deve ser colocada fora do `Loop` , pois a cada vez que o programa terminar a execução do `Loop` ela voltará a possuir o valor `0` , mais adiante você entenderá isso melhor.

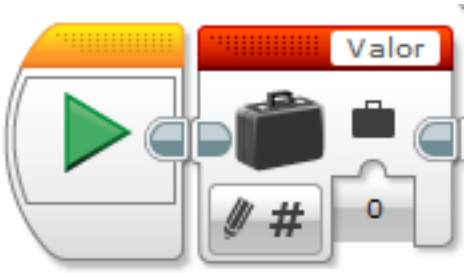



Figura 20.24: Solução da questão 1 - Variável "Valor"

A próxima variável, que será chamada de `Contador` , será responsável por armazenar as detecções da luz negra pelo sensor. Vamos utilizá-la para realizar a comparação do valor armazenado com o número limite de detecções que determina o enunciado. Com esta variável, realizamos o controle do número de execuções do `Switch` . Devemos realizar a conexão entre as variáveis, o valor da variável `Valor` deve ser escrito na variável `Contador` . Com isso, será criada a lógica responsável por executar por 3 vezes o ciclo anterior do robô após as 2 detecções da cor escura.

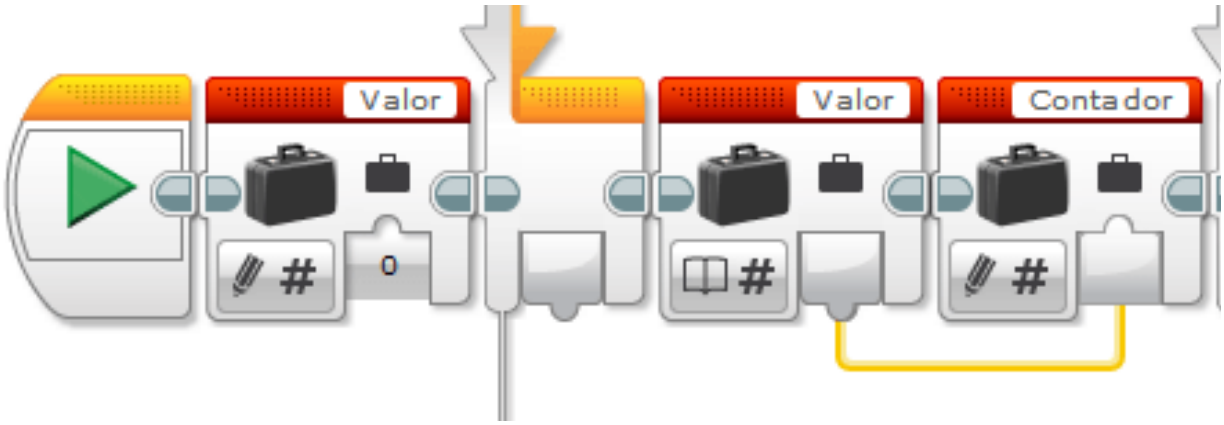



Figura 20.25: Solução da questão 1 - Variáveis: "Contador" e "Valor"

O próximo `Loop` se trata do programa *Varia o do Rob Sum* , aquele que termina a execução do programa após as duas detecções da cor escura. A próxima etapa é programá-lo para que, quando as duas detecções sejam feitas, o robô pare por 5 segundos e volte a executar o ciclo por 3 vezes.

Para tal, vamos adicionar a variável `Valor` no modo de leitura ( `Read` ). Em seguida, enviamos o seu valor à entrada `a` do bloco `Compare` , que deve ser configurado no modo `Equal To` . Na entrada `b` , vamos escrever o número `0` . O resultado da operação feita pelo bloco `Compare` enviamos ao `Switch` , que devemos configurar no modo `Logic` .

No começo do programa, escrevemos o valor `0` na variável `Valor` . Até o momento, após a execução do `Switch` , a variável continua com o valor zero. Mas, agora, construiremos no `Switch` a lógica para que o programa repita o ciclo anterior por três vezes.

Dentro do `Switch` , adicionaremos a variável `Valor` duas vezes. A primeira variável será adicionada no `Case True` do `Switch` , e escreveremos o valor `-1` nela. A outra variável `Valor` será adicionada no `Case False` , nela escrevemos o valor `0` . O `Switch` ficará da seguinte maneira:

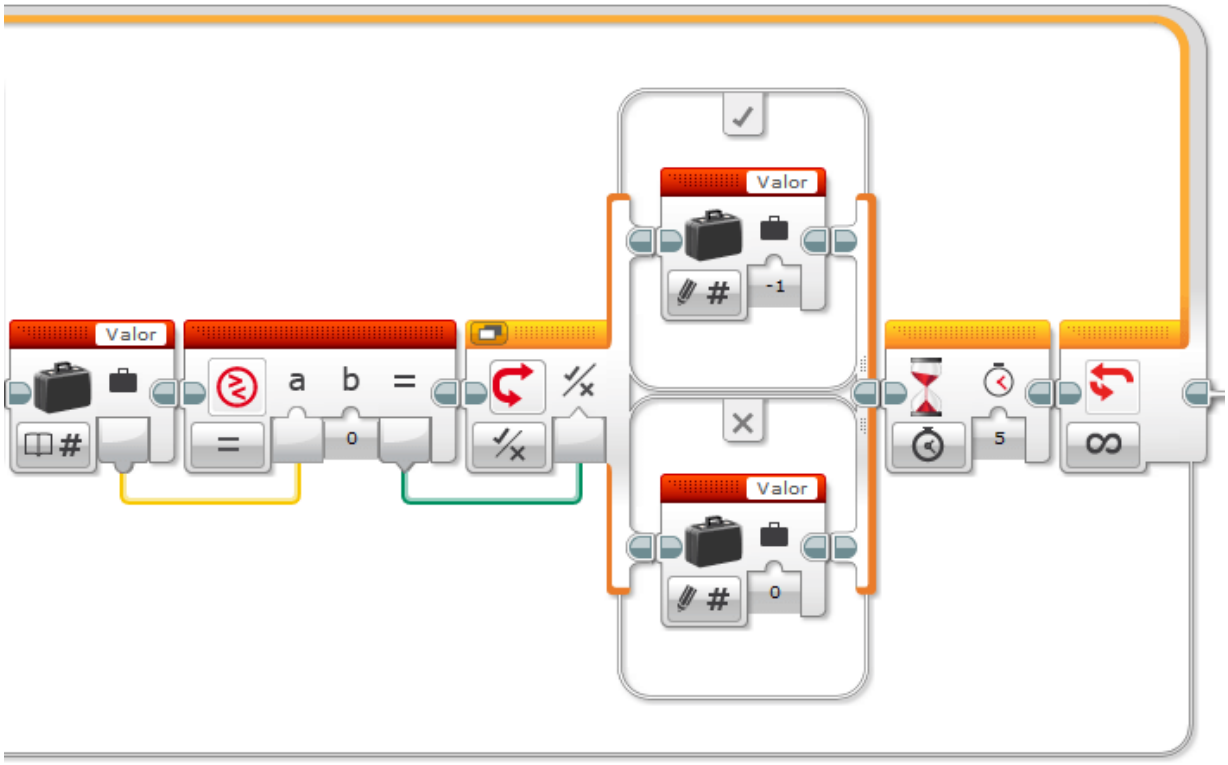



Figura 20.26: Solução da questão 1 - Parte final do código

Agora, vamos analisar a lógica por trás do código.

Nos blocos anteriores ao `Switch` fizemos uma comparação para que, quando o valor da variável `Valor` seja zero, isso resulte em um valor lógico `True` . Para o valor `True` escreve-se na variável o valor `-1` , para `False` o valor `0` .

Em seguida, está o bloco `Wait` no modo `Time Indicator` , configurado para que o robô espere por 5 segundos.

Vamos analisar mais a fundo a situação, para isso, simularemos execuções do programa e observaremos os valores das variáveis.

Ao declarar o valor `-1` na variável `Valor` , esse valor será escrito também na variável `Contador` . Relembrando e exemplificando:

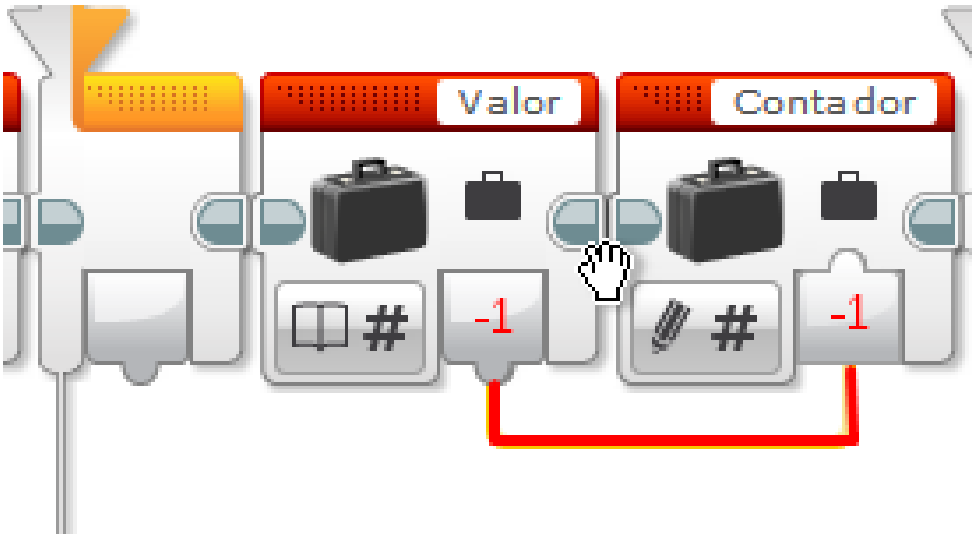



Figura 20.27: Solução da questão 1 - Demonstração do armazenamento de valores nas variáveis

Com o valor de `-1` , serão necessárias três detecções para que a variável possua valor igual a 2.

1ª execução: Contador (= -1) + 1 (valor adicionado pelo parâmetro `b` ) = 0

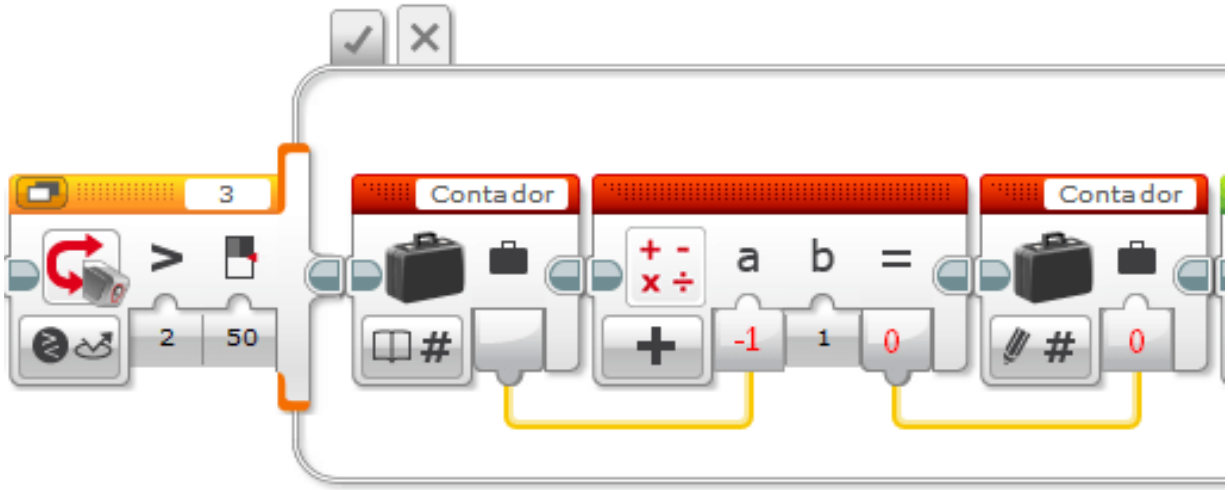



Figura 20.28: Solução da questão 1 - Demonstração da primeira execução

## 2ª execução: Contador (= 0) + 1 = 1

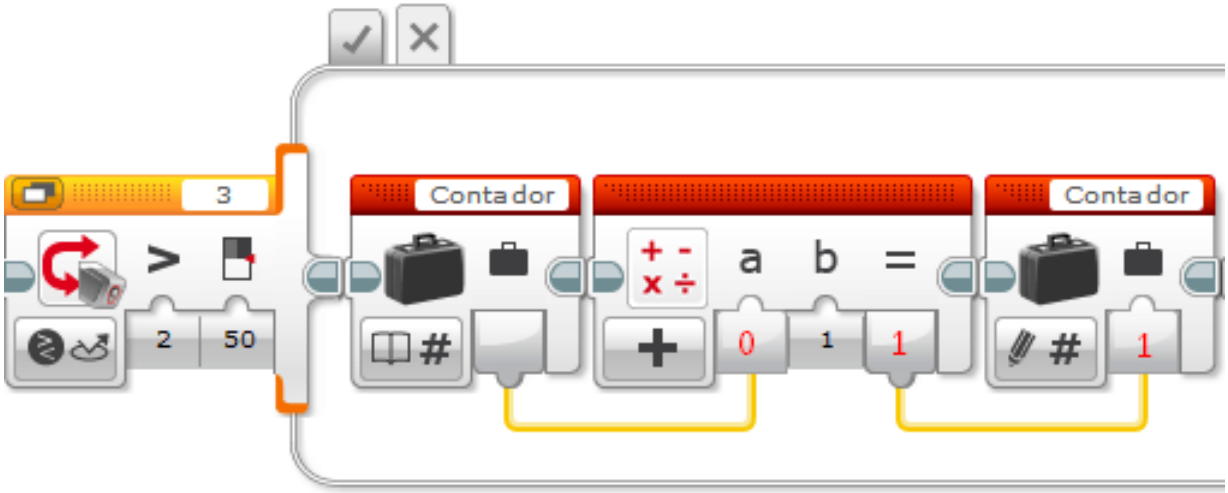



Figura 20.29: Solução da questão 1 - Demonstração da segunda execução

## 3ª execução: Contador (= 1) + 1 = 2

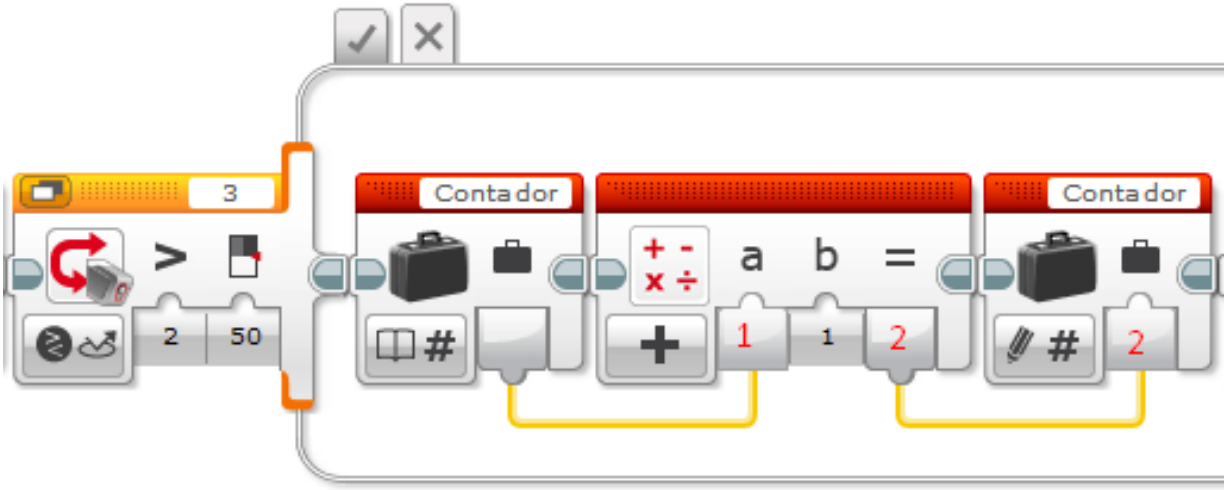



Figura 20.30: Solução da questão 1 - Demonstração da terceira execução

Após três detecções, o `Loop` interno é interrompido e o programa volta ao `Loop` externo. Agora a variável `Valor` será lida novamente. Desta vez, a variável apresenta o valor `-1` , resultando em `False` . Para o valor `False` , é escrito o valor `0` na variável `Valor` , desta forma volta-se ao início do programa, como se estivesse executando-o pela primeira vez.

**2.**

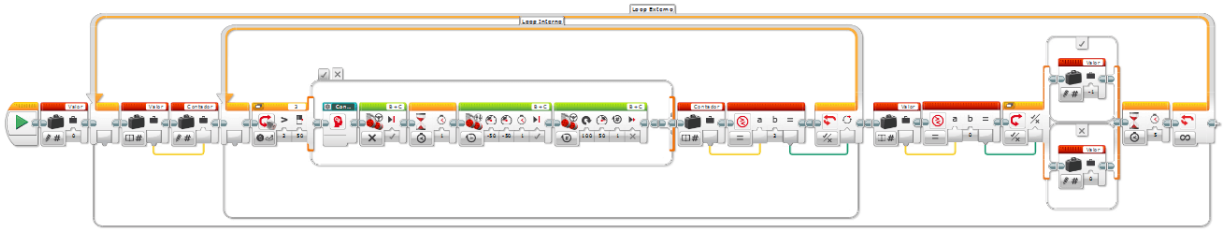


Figura 20.31: Robô Sumô: Exercício 2 - Programa completo

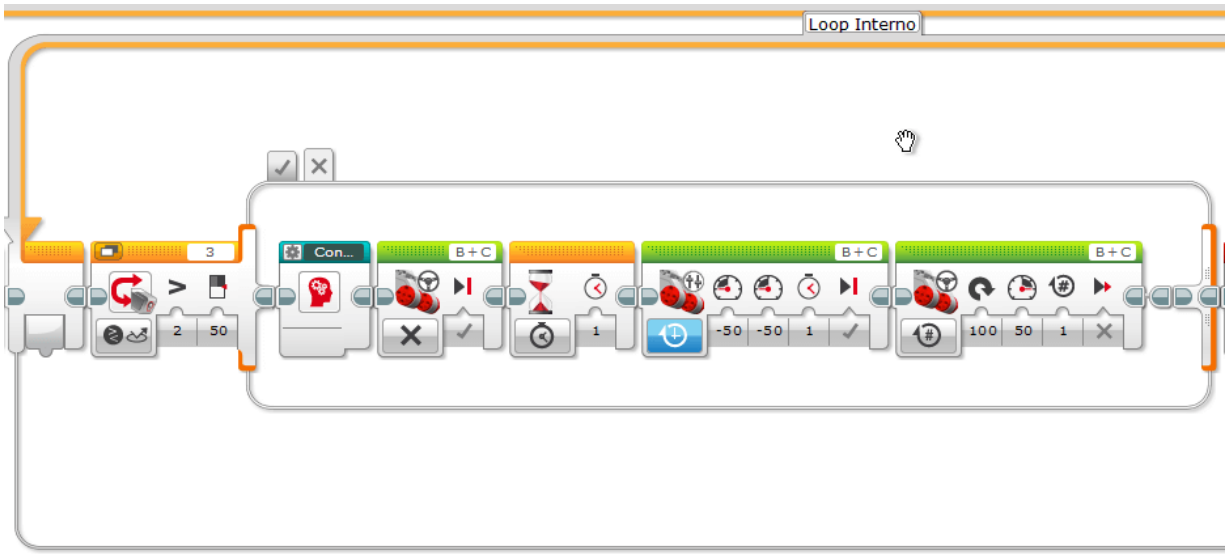



Figura 20.32: Robô Sumô: Exercício 2 - Detalhe - My Block

Para este exercício, vamos utilizar a ferramenta `My Block Builder`. Com ela, criaremos um novo bloco com os blocos que realizam a contagem.

Este exercício é muito simples de ser resolvido. Basta que você siga os passos descritos na seção *My Blocks*, que acabamos de ver, e aplicar ao exercício.

Veja como ficou o programa com o bloco próprio criado:

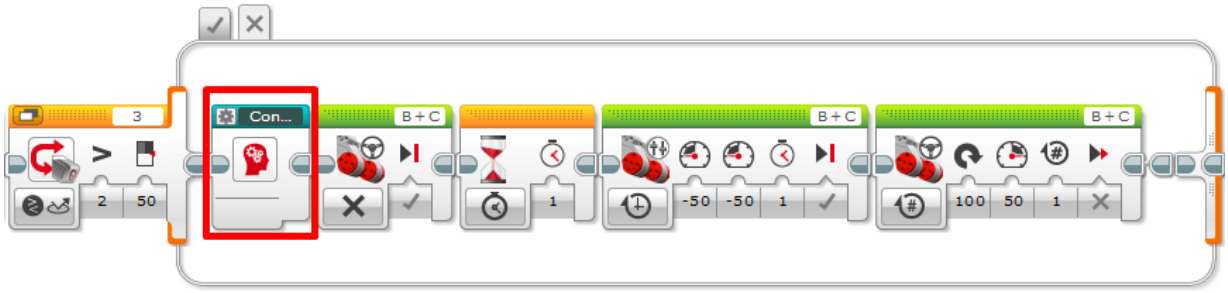



Figura 20.33: Solução da questão 2 - Programa utilizando o My Blocks

**3.**

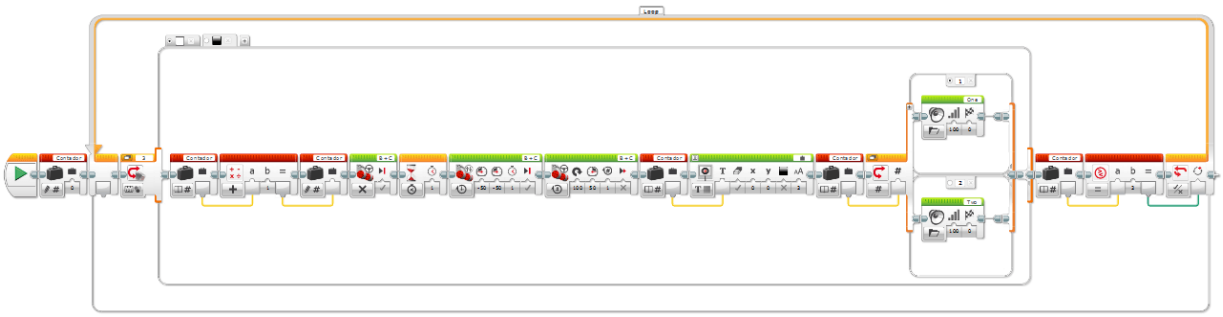


Figura 20.34: Robô Sumô: Exercício 3 - Programa completo

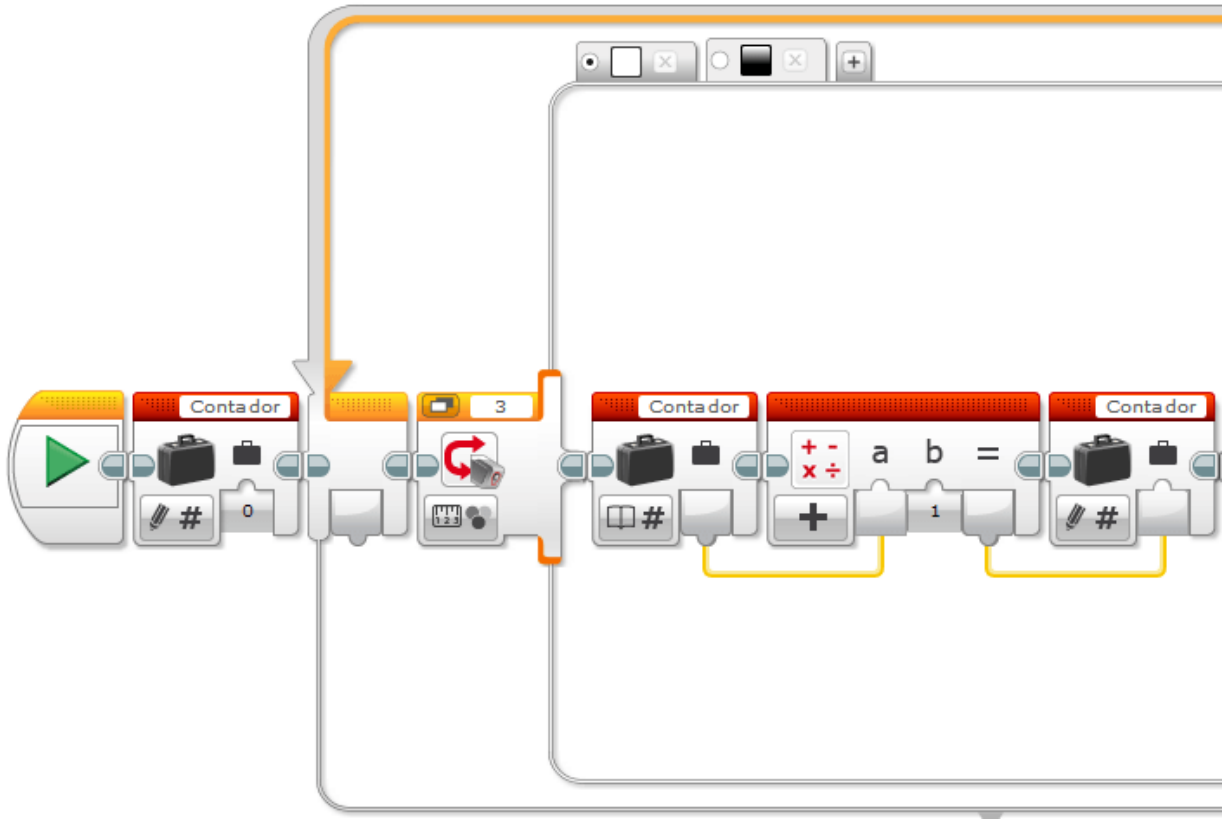



Figura 20.35: Robô Sumô: Exercício 3 - Detalhe 1

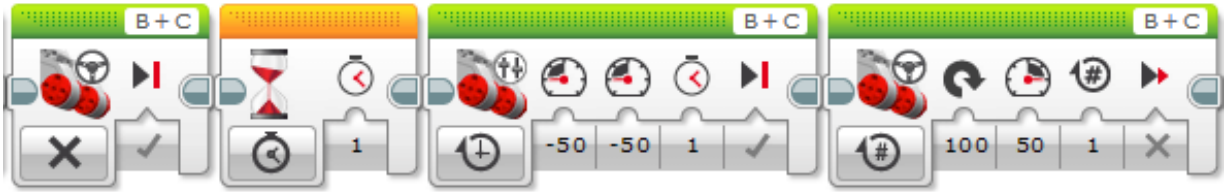



Figura 20.36: Robô Sumô: Exercício 3 - Detalhe 2

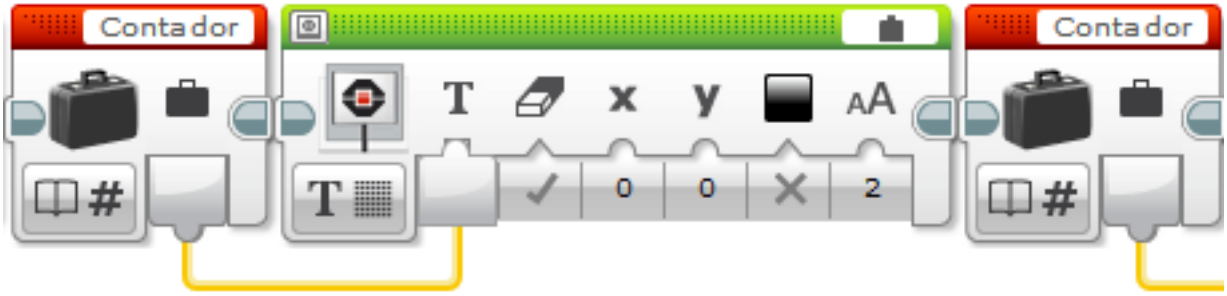



Figura 20.37: Robô Sumô: Exercício 3 - Detalhe 3

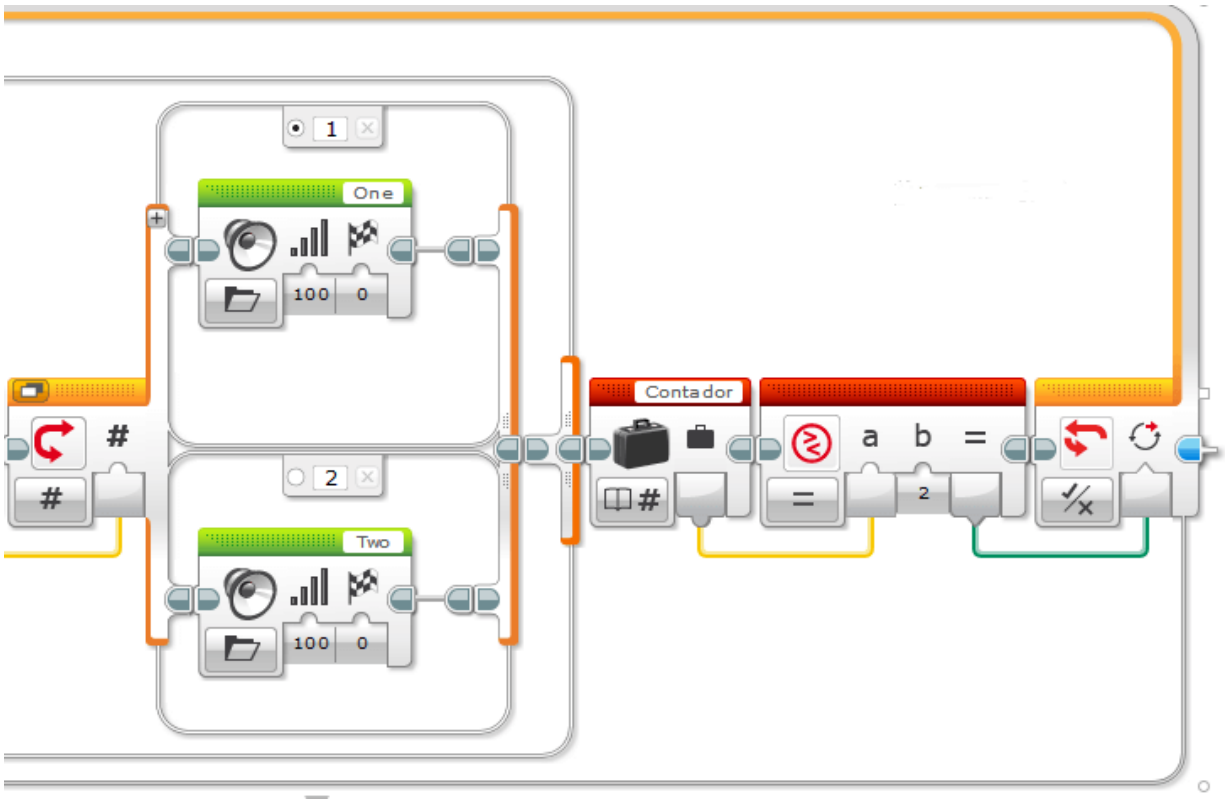



Figura 20.38: Robô Sumô: Exercício 3 - Detalhe 4

Neste exercício, utilizaremos o `Switch` no modo `Colour Sensor` configurado em `Measure - Colour` . Vamos adicionar dois casos para ele, um para a detecção da cor branca e outra para a detecção da cor negra.

A cor branca representa a luminosidade maior que 50, e a cor negra luminosidade menor 50.

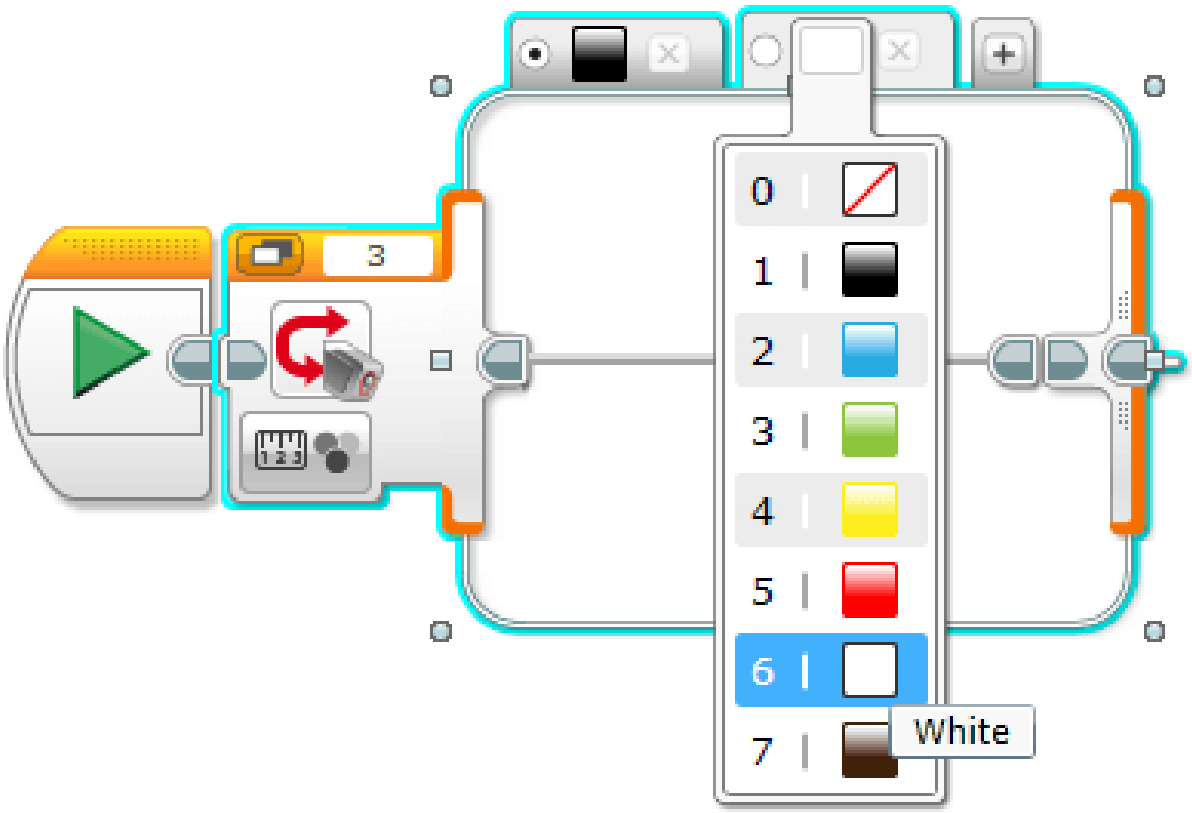



Figura 20.39: Solução da questão 3 - Definindo os Cases

O caso `White` :

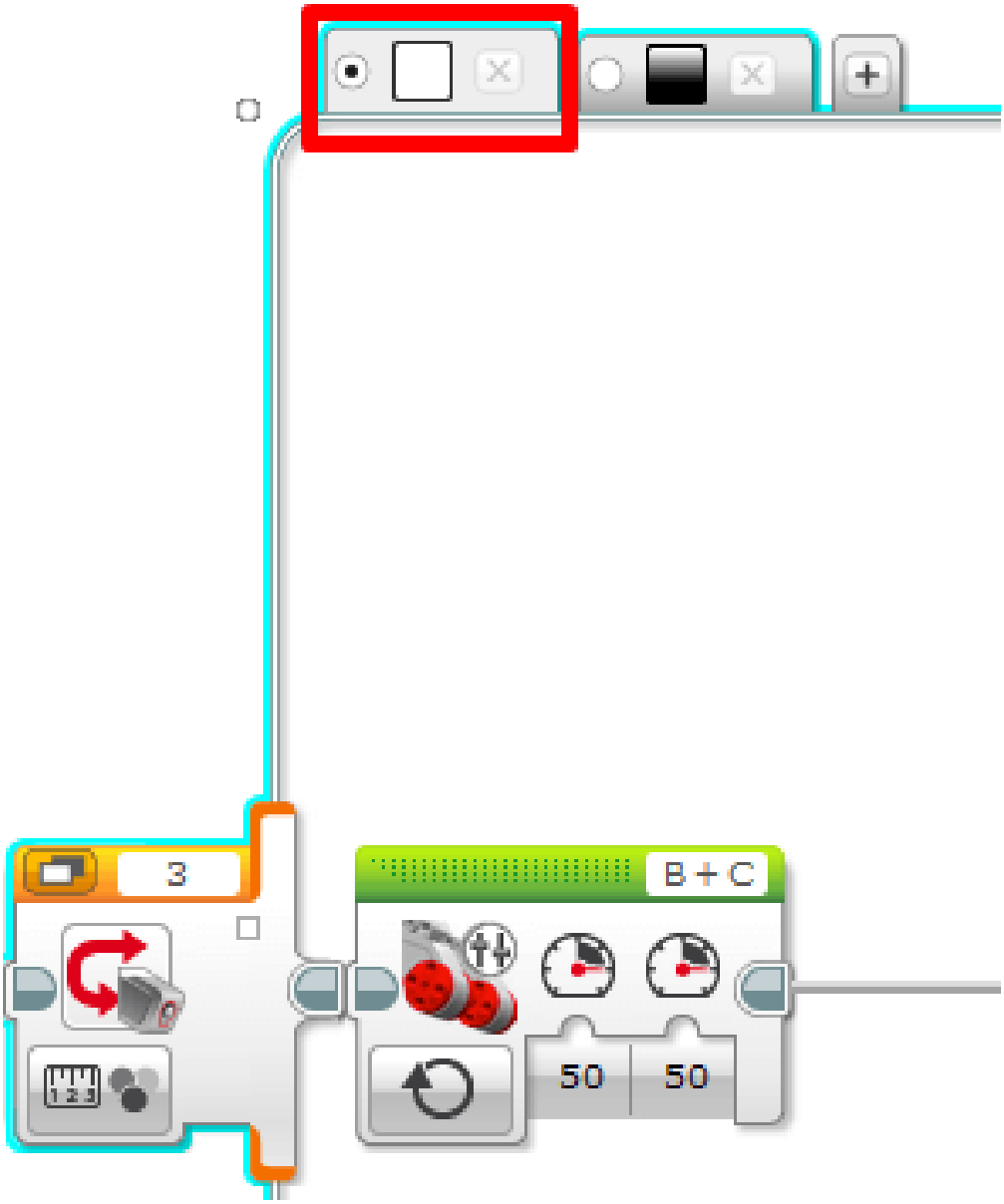



Figura 20.40: Solução da questão 3 - Sequência de blocos contidas no Case White

Para o caso `Black` , adicionamos os mesmos blocos utilizados nos primeiros programas, porém, aqui também faremos a parte responsável por exibir do número de detecções no display do Brick e gerar sons que correspondam ao número de detecções.

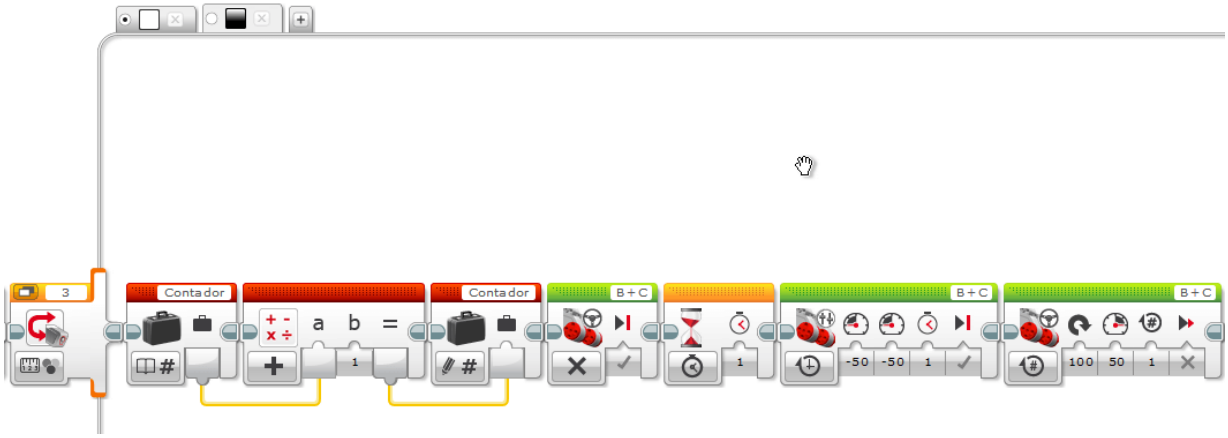



Figura 20.41: Solução da questão 3 - Sequência de blocos contidas no Case Black

Após realizado o incremento na variável e executados os movimentos, vamos exibir o número de detecções na tela. Fazemos isso adicionando a variável `Contador` no modo de leitura ( `Read` ) e enviando o seu valor à entrada `Text` do bloco `Display` , que deve ser configurado em `Wired` .

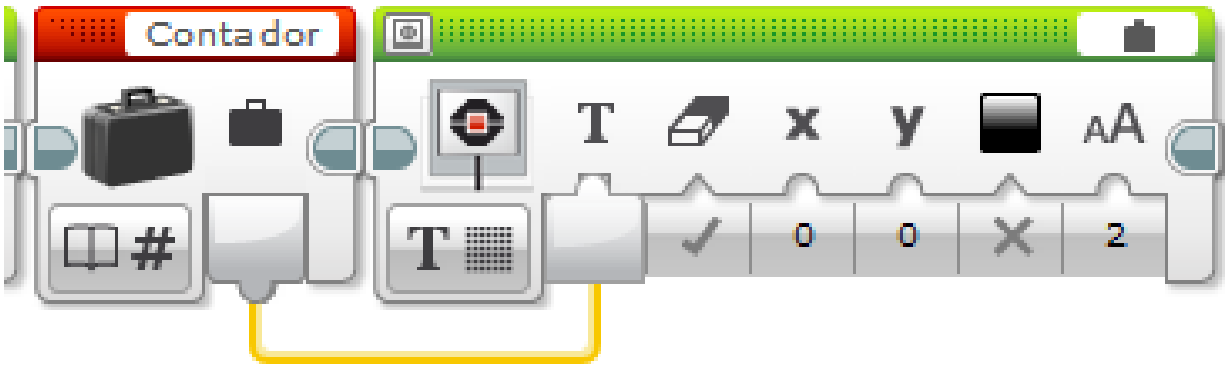



Figura 20.42: Solução da questão 3 - Enviando o valor da variável "Contador" ao bloco Display

Em seguida, o robô vai gerar o som corresponde ao número da execução. Para essa operação, novamente será realizada a leitura do valor da variável `Contador` . Primeiro, vamos adicionar a variável `Contador` e configurá-la no modo `Read` . Após isso, vamos adicionar um

`Switch` . Dessa vez, utilizaremos um `Switch` configurado no modo `Numeric` . Agora, com esses dois blocos adicionados, realizaremos a comunicação entre eles. Para isso, conectamos a saída `Value` da variável `Contador` à entrada `Number` do `Switch - Numeric` .

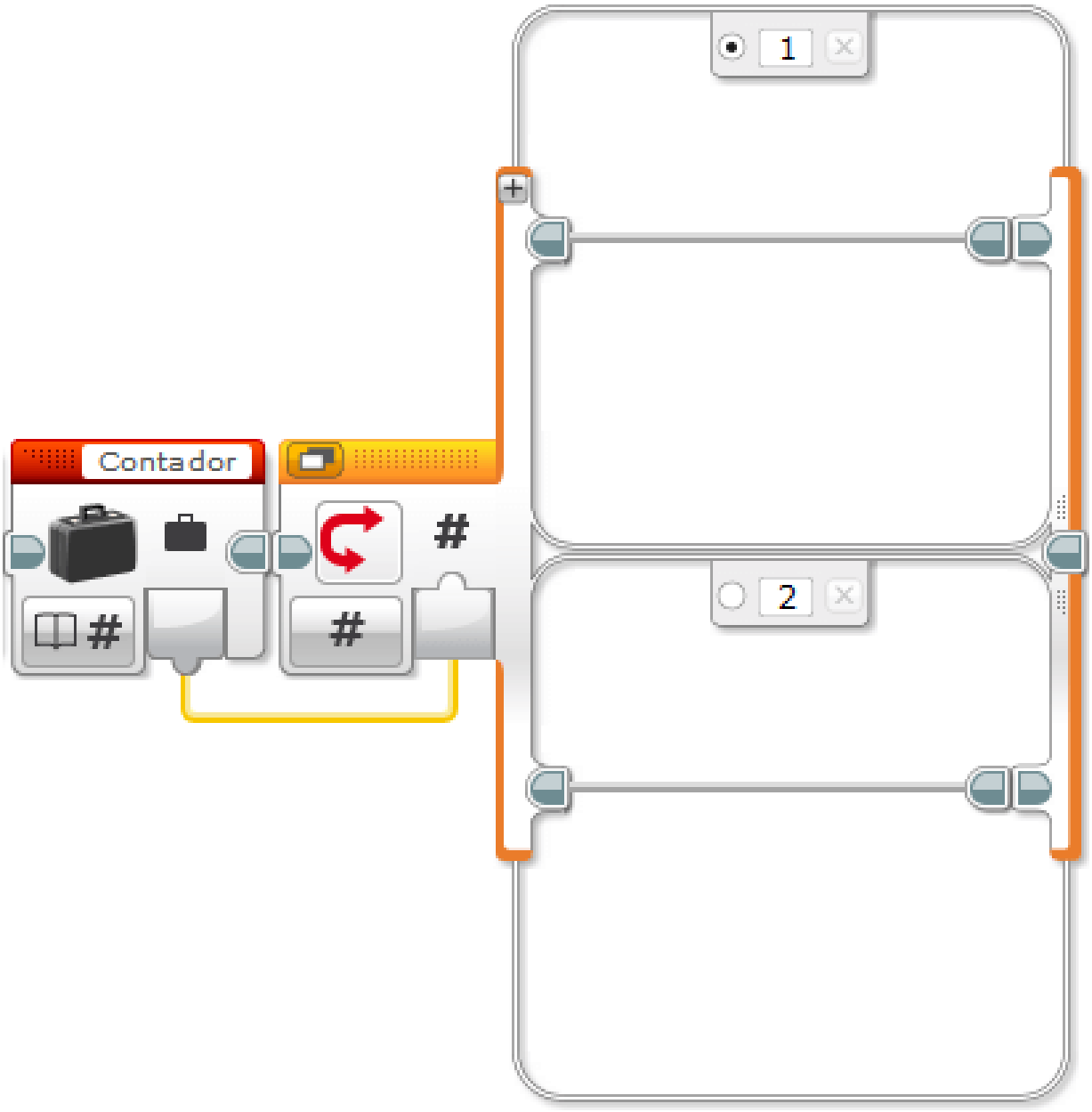



Figura 20.43: Solução da questão 3 - Variável "Contador" e Switch Numeric

É necessário definir os casos 1 e 2 para o `Switch` . O caso 1 corresponde à primeira execução do `Switch` , nele adicionamos o

bloco Sound . No bloco Sound , vamos escolher o modo Play File e, em Numbers , selecionamos o arquivo One . Para o caso 2 faremos o mesmo, porém, aqui selecionamos o arquivo Two . Desse modo, caso o valor da variável seja igual a 1, será reproduzido o som One (Um), e caso seja 2 o som Two (Dois).

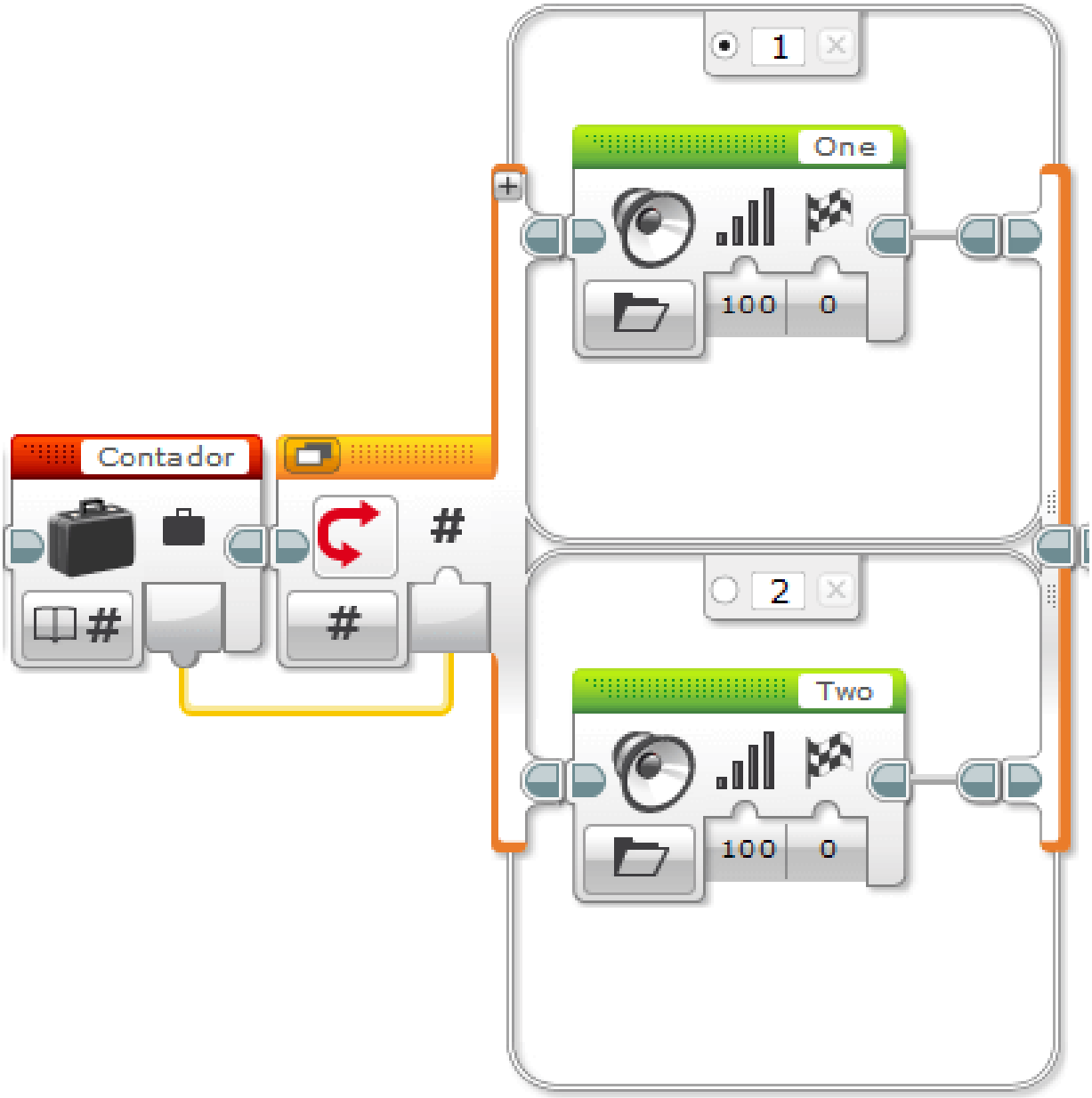



Figura 20.44: Solução da questão 3 - Switch Numeric com blocos de som (Sound)

Após o `Switch` , temos os blocos responsáveis pelo controle do `Loop` , que são os mesmos apresentados no programa *Varia  o do Rob Sum* .

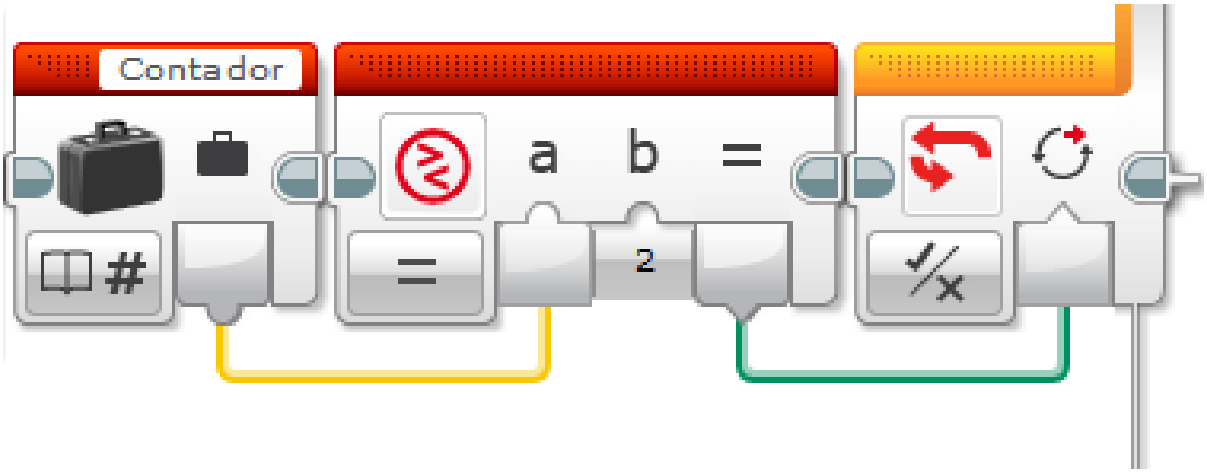



Figura 20.45: Robô Sumô - Blocos responsáveis pelo controle do Loop

**Pronto! E agora?**

Trabalhamos neste capítulo mais um robô e, a partir dele, vimos novos conceitos e novas lógicas. Partindo de um programa simples, criamos uma variação mais elaborada do programa, com mais funcionalidades. Na seção de exercícios, fornecemos enunciados cada vez mais completos e rebuscados. A ideia principal dos exercícios é propor desafios para que você possa desenvolver sua lógica de programação e suas habilidades.

Vimos uma importante ferramenta, o `My Block Builder` que, conforme você percebeu, é uma ferramenta de enorme utilidade. A criação de blocos próprios nos dá um grande ganho de tempo, pois assim não precisamos reescrever sequências inteiras de códigos. Além disso, ele garante uma grande otimização no tamanho e organização, já que com ele encapsulamos grandes sequências de blocos em um só bloco.

# Capítulo 21
# Explorer

Neste capítulo, vamos programar um robô que terá a seguinte missão: explorar. Nosso robô será capaz de percorrer todo um perímetro, livrando-se de obstáculos que se apresentarem durante o caminho, e escolher o melhor caminho para que possa prosseguir com seu trajeto.

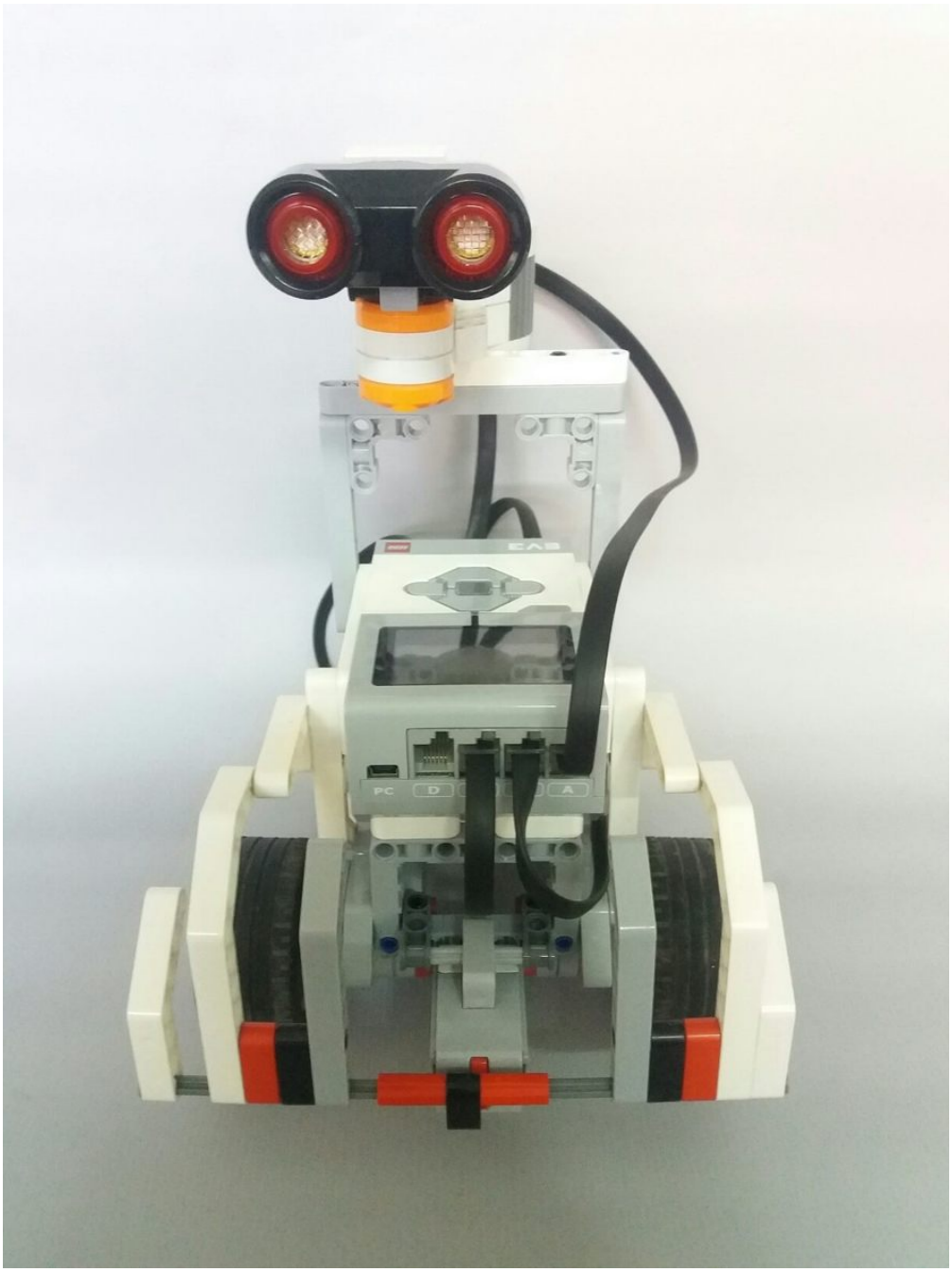


Figura 21.1: Explorer

## Instruções de montagem

Para a montagem do robô, vamos utilizar a mesma base de antes. Porém, faremos as modificações necessárias para que tenhamos o robô explorador. Na pasta `Instructions` referente a esta experiência

estão as instruções para que você monte seu robô. Você pode acessá-la diretamente pelo link:

https://github.com/vitor000x/ev3_Codes/tree/master/Experiencias/3-Explorer/Instructions.

Para esta experiência, certifique-se de que o motor da esquerda está conectado na porta B e o motor da direita está conectado na porta C. Isto é apenas uma padronização, dessa forma ficará mais fácil de acompanhar e entender o código.

**Objetivo do programa**

O programa que será criado tem a função de comandar um robô que explora um cômodo, casa ou qualquer lugar onde ele estiver. Ele tenta “enxergar” obstáculos através do sensor ultrassônico. Este ato é realizado pela cabeça do robô que, logo após detectar algum objeto, gira para a direita e para a esquerda para escolher o melhor caminho para prosseguir a exploração. Assim, pode-se garantir que o robô continue sua missão exploratória, sem que nenhum obstáculo detectável pelo sensor ultrassônico o interrompa.

No caso de acidentalmente o robô colidir com algum obstáculo que estiver fora do seu alcance de visão, isso acionará o sensor de toque que fará com que o robô pare e decida em que direção seguir.

O veículo deve parar de se movimentar quando estiver a uma distância menor do que 20 centímetros de um alvo. A distância será medida pelo sensor ultrassônico, posicionado sobre a cabeça do robô (e acoplado a um motor, que permite seu giro). O programa será escrito de forma que permita ao veículo, após sua parada, alterar seu rumo para a direção (esquerda/direita) que apresente a maior distância entre o robô e um obstáculo.

Ao girar para a direção que apresenta uma maior distância com um possível obstáculo, o robô está escolhendo a melhor trajetória momentânea possível. As medições feitas no lado esquerdo e direito, logo após a detecção de um obstáculo, antecipam um

possível obstáculo que o robô encontraria caso simplesmente girasse para um lado aleatório.

Para que nosso robô possa executar a sequência passos infinitamente, o primeiro bloco que adicionaremos após o bloco `Start` será o bloco `Loop` configurado no modo `Unlimited` . Nosso robô tem a missão de explorar o perímetro, por isso vamos adicionar o bloco `Move Tank` utilizando potência média.

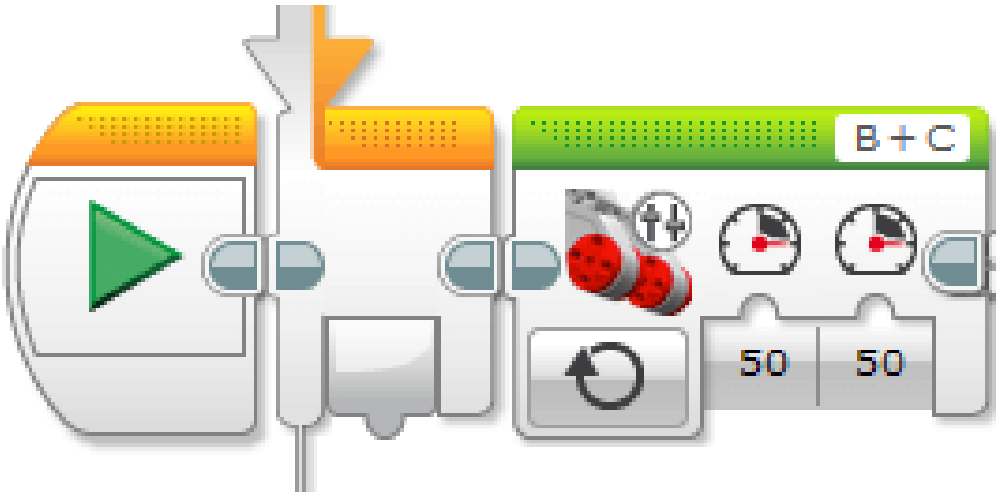



Figura 21.2: Primeiros blocos (após o bloco Start): Loop e Move Tank

Até agora, o nosso robô apenas anda para a frente, logo, caso ele encontre um obstáculo talvez não conseguirá continuar o trajeto; pois seu caminho poderá estar bloqueado por um obstáculo que impeça ou atrapalhe sua passagem. Por isso, inseriremos um `Switch` , que conterá a sequência de blocos contendo as ações que serão tomadas pelo robô quando um obstáculo for detectado.

O `Switch` que utilizaremos é do tipo sensor e vamos configurá-lo da seguinte maneira: `Ultrasonic Sensor – Compare – Distance Centimetres` . Esse `Switch` será responsável por quase todas as ações do programa, realizará a detecção de obstáculos, medirá as distâncias e, então, calculará a menor distância.

Agora temos de realizar as configurações necessárias no `Switch` para que o programa funcione de maneira correta. No parâmetro `Compare`

`Type` , selecionamos o tipo de comparação `Less Than` (Menor que). Em `Threshold Value` ajustamos o valor limiar que, conforme pediu o enunciado, deve ser de 20 cm. Isso significa que, quando detectar um objeto em uma distância menor que 20 centímetros, ele entra em ação e executa os comandos para o `Case True` , caso o valor seja maior que 20, ele executa os comandos para o `Case False` .

Vamos analisar a composição deste `Switch` : no `Case False` não inseriremos nenhum bloco, pois, caso nenhum obstáculo seja detectado a uma distância menor que 20 centímetros, nosso robô simplesmente continua andando.

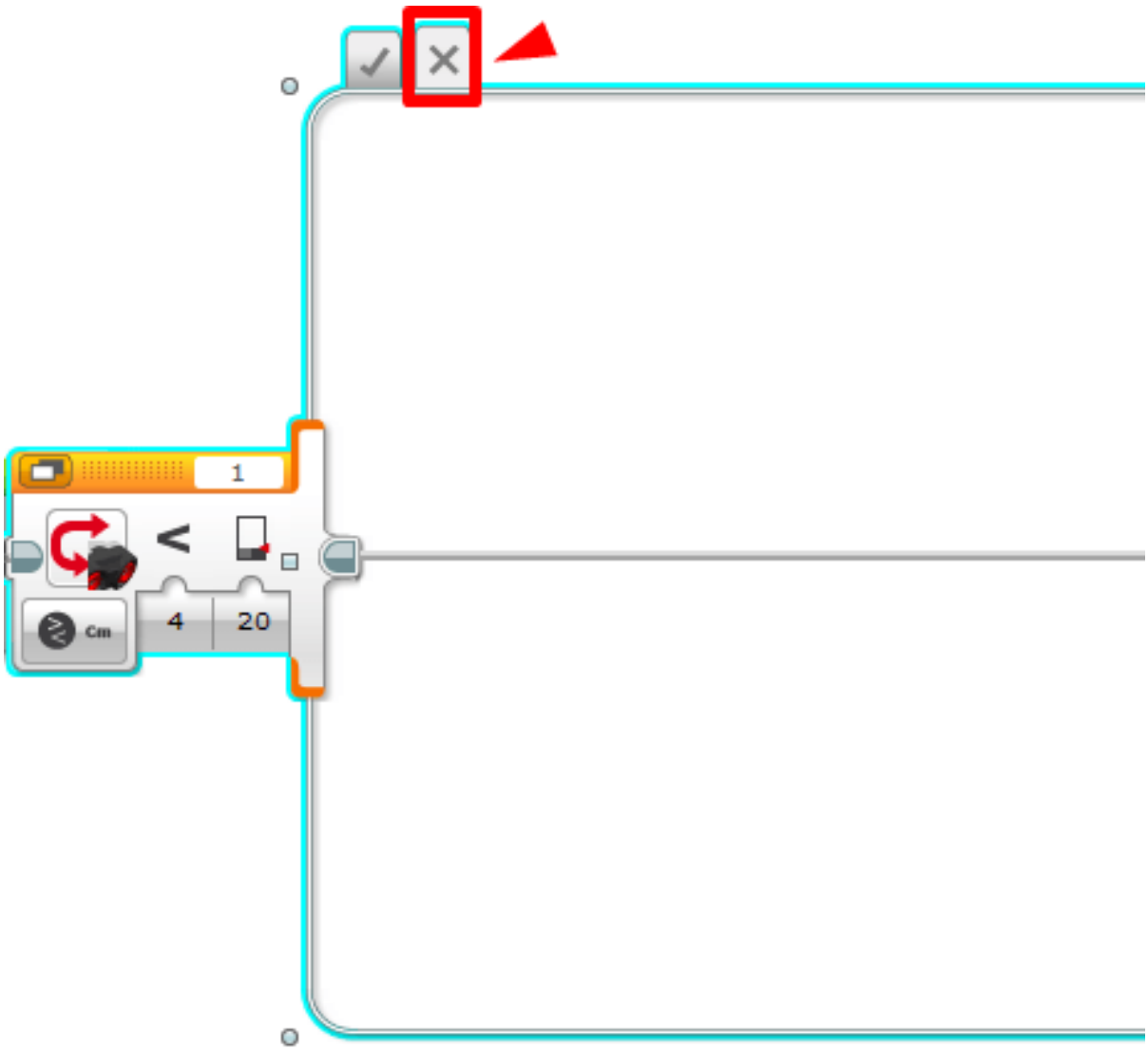



Figura 21.3: Switch - Ultrasonic Sensor - Compare - Distance in Centimetres: Case False

Agora podemos passar para o `Case True` do `Switch` . Quando o programa executa o `Case True` do `Switch` é porque foi detectado um objeto a uma distância menor que 20 cm, logo, para que o robô não bata no obstáculo, é preciso cessar seu movimento retilíneo. Para isso, basta inserir o bloco `Move Tank` no modo `Off` , assim, o robô cessará seu movimento imediatamente. Após essa espera, inseriremos o bloco responsável pelo movimento de marcha ré do robô que entrará em ação quando um obstáculo for detectado. Para isso, utilizaremos novamente o bloco `Move Tank` configurado no modo `On for Seconds` , para executar por 1 segundo, e ajustaremos suas

potências com valores negativos, desse modo, o movimento será executado em sentido oposto.

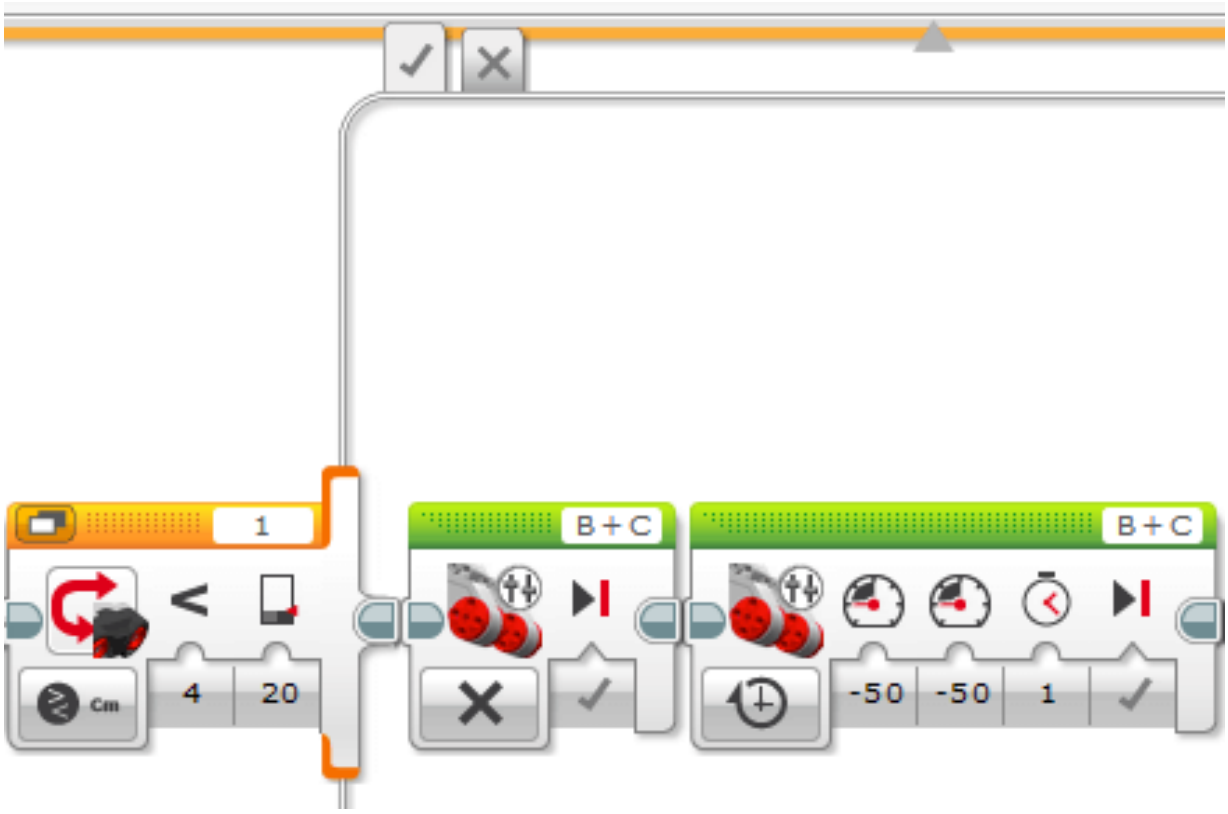



Figura 21.4: Switch - Ultrasonic Sensor - Compare - Distance in Centimetres: Case True

Depois que o robô detectar o obstáculo e parar, o robô deve realizar as medições, para “saber” para qual direção prosseguir. O primeiro bloco `Large Motor` que adicionaremos será responsável pelo giro à direita, por isso, vamos configurá-lo no modo `On for Degrees` e ajustá-lo para rotacionar 90 graus (movimento para direita). Logo após a rotação, o sensor ultrassônico acoplado ao motor realizará a medição. Para que isso seja feito, vamos adicionar o bloco `Ultrasonic Sensor` com a seguinte configuração: `Measure - Distance in Centimetres` .

**Observação**

Neste programa foi usado um `Large Motor` (pois na montagem do robô foi utilizado o motor do kit anterior, o NXT, que, conforme foi dito no início do livro, funciona com o kit EV3), mas poderia ter sido utilizado também o próprio `Medium Motor` que acompanha o kit EV3. No último caso, você apenas precisaria realizar algumas modificações na construção do robô.

O próximo passo é realizar o movimento para a esquerda. Para isso, devemos configurar o bloco `Large Motor` no mesmo modo, porém vamos configurar o parâmetro `Degrees` com -180 graus, pois ele deve voltar à posição original (centro, que corresponde a -90 graus) e aí sim girar para a esquerda (que corresponde aos outros -90 graus). Em seguida, adicionamos outro bloco sensor, (com as mesmas configurações de antes) para que possa ser aferida a medição da distância.

Depois de realizada a medição do lado esquerdo, o sensor deve voltar à sua posição original. Para tal, utilizaremos outro bloco motor, no mesmo modo. Neste bloco motor, ajustaremos seu parâmetro `Degrees` com 90 graus.

Veja a representação detalhada dos movimentos:

- 90º (primeiro movimento do motor A, para a direita)
- -180º (segundo movimento do motor A, para a esquerda)
- = -90º (sensor já no lado esquerdo)
-
    - 90º (terceiro movimento do motor A, fazendo com que o motor e, por consequência, o sensor retornem a posição central)
- = 0º (o sensor retorna à posição central)

Figura 21.5: Blocos responsáveis que realizarão o giro do motor (no qual o sensor ultrassônico está acoplado) e medida das distâncias

Para que a comparação possa ser feita, utilizaremos o bloco matemático `Compare` . Enviaremos os dados obtidos pelas medições do sensor ultrassônico (que se encontram na saída `Distance in Centimetres` do bloco `Ultrasonic Sensor` ) para as entradas do bloco `Compare` . Nele, escolhemos a comparação `Greater Than` (Maior que), pois queremos saber qual lado possui a maior distância a um obstáculo. Conectamos o `Data Wires` responsável pela medição no lado direito na entrada `a` do bloco `Compare` , e o `Data Wires` contendo o resultado da medição no lado esquerdo vai na entrada `b` do bloco. Se a distância do lado direito for maior que a do lado esquerdo, isso resulta em um resultado lógico verdadeiro.

**a (direita) > b (esquerda) = True**

**a (direita) < b (esquerda) = False**

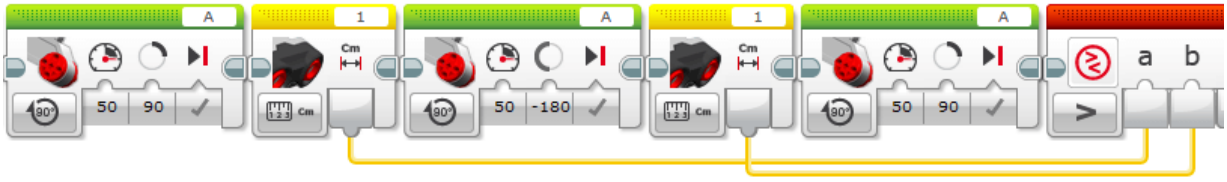



Figura 21.6: Bloco Compare recebendo os valores medidos pelo sensor por meio do Data Wires

Enviaremos o resultado dessa comparação a um `Switch - Logic` que executará a ação correspondente. Adicionamos o bloco `Switch` configurado no modo `Logic` e, então, ligamos a saída `Result` do bloco `Compare` à entrada `Logic` do `Switch` . Nosso último passo é inserir nos

Cases do Switch as ações correspondentes ao valor lógico obtido por meio do bloco Compare .

Conforme vimos, caso a distância medida no lado direito seja maior que a distância medida no lado esquerdo, teremos um resultado lógico True . Por isso, no Case True do Switch inseriremos o bloco que movimentará o robô para a direita. Utilizaremos o bloco Move Steering configurado no modo On for Seconds . É no parâmetro Steering que inserimos o valor de rotação, nele inseriremos o valor 35 . No parâmetro Seconds , deixamos com o valor 1 , pois um segundo é suficiente. No parâmetro Brake at End selecionamos a opção False que corresponde ao Coast , com o qual o movimento de parada do motor é mais suave.

O valor 35 utilizado foi obtido por meio de testes, tendo em vista que quanto mais próximo de 100 mais fechado é o giro. A definição de “x” graus, ou então, “y” rotações, diz respeito apenas ao motor; ou seja, a quantidade em graus ou rotações que o motor girou. Por exemplo, adicionar um bloco Move Steering no modo On for Degrees , configurando o parâmetro Degrees com o valor 360, não significa que o robô efetuará um giro de 360 graus, quem gira 360 graus é o motor. Fatores externos influenciam na direção do robô como, por exemplo, o espaçamento entre as rodas, diâmetro das rodas e outros.

O movimento responsável pelo giro à direita já está pronto. Criaremos agora o movimento para a esquerda. Para isso, basta inserirmos no Case False do Switch o mesmo bloco motor, porém, no parâmetro Steering , utilizaremos um valor negativo.

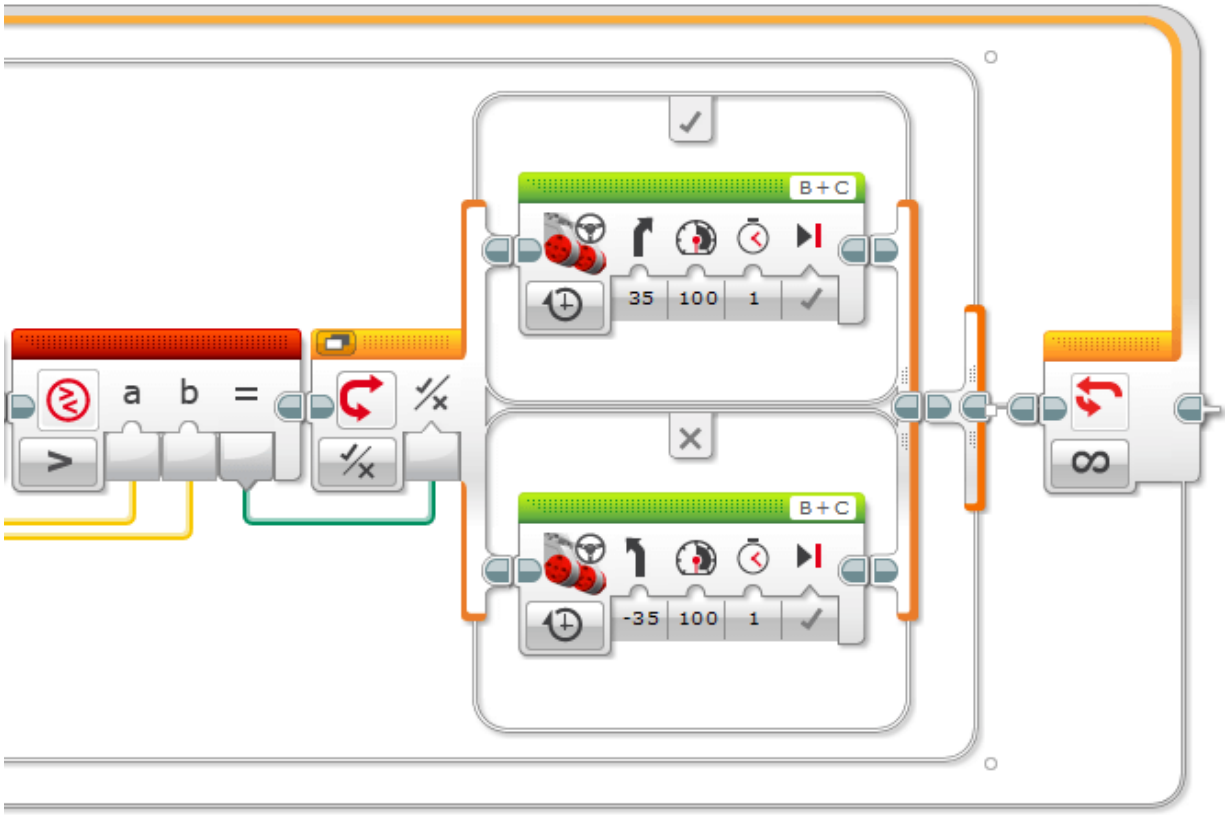



Figura 21.7: Switch Logic contendo os blocos referentes ao giro à direita e à esquerda

Este foi o último passo, nosso robô já está pronto para uso, veja como ficou o programa completo:

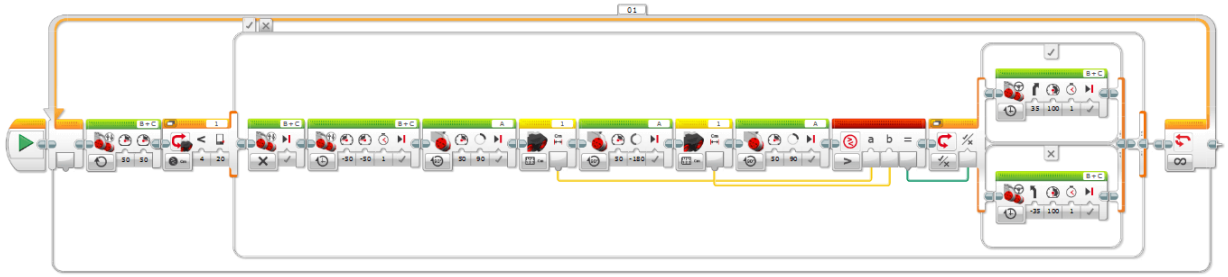


Figura 21.8: Robô Explorer - Código Completo

# 21.1 Variações do Robô Explorer

**1.**

Considere o programa que acabamos de fazer. Vamos acrescentar a ele a seguinte restrição: o robô também parará sempre que colidir com obstáculos inferiores, detectáveis somente pelo sensor de toque.

Para isso, faremos alguns incrementos no código, mas, também, serão realizadas alterações na configuração de alguns blocos já dispostos no programa anterior. O `Switch` externo agora será do tipo lógico, pois deve atender ao sensor ultrassônico e ao sensor de toque. Antes, como utilizávamos apenas o sensor ultrassônico, logo, bastava configurar `Switch` no modo `Ultrasonic Sensor` . Porém, agora, precisamos que o `Switch` seja acionado não só se um objeto estiver a uma distância menor que 20cm, mas, também, caso o sensor de toque seja pressionado.

Utilizaremos a configuração `Logic` para o `Switch` , pois, por meio de uma operação lógica (que veremos a seguir), será enviado a ele um resultado lógico. Por meio desse resultado, recebido pelo `Data Wires` , o `Switch` saberá qual `Case` executar.

Para isso são utilizados os blocos sensores e o bloco matemático `Logic Operations` . O bloco `Logic Operations` deverá ser configurado no modo `Or` , isso significa que, caso um evento e/ou outro aconteçam, um valor lógico `True` será obtido. Com essa configuração, o robô será capaz de se livrar dos obstáculos detectados tanto pelo sensor ultrassônico, quanto pelo sensor de toque.

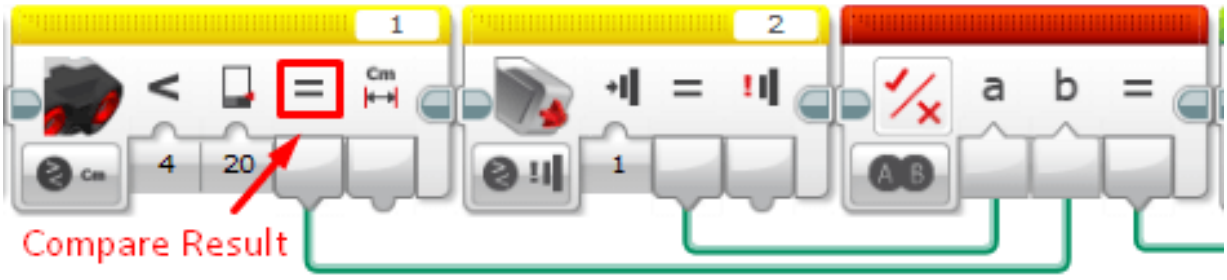



Figura 21.9: Robô Explorer - Variação 1: Blocos sensores e bloco Logic Operations

A configuração do sensor ultrassônico é a mesma. Configura-se o sensor de toque (*Touch Sensor*) no modo `Compare – State` , no parâmetro `State` , escolhemos o estado `Pressed` , representado pelo número 1. Com essa configuração, quando o sensor de toque estiver pressionado, um valor lógico verdadeiro será enviado através do `Data Wires` ao bloco `Logic Operations` (que já foi estudado anteriormente).

Para isso, devemos conectar o `Data Wires` contido nas saídas dos blocos sensores ( `Compare Result` ) e enviar às entradas do bloco `Logic Operations` . O `Result` do bloco lógico é então enviado ao `Switch` .

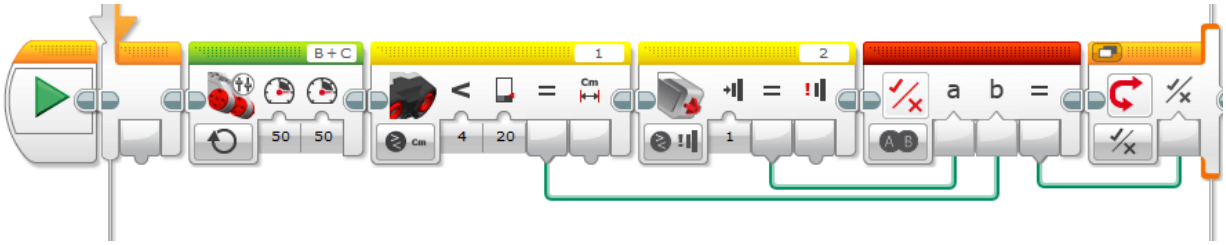



Figura 21.10: Robô Explorer - Variação 1: sequência lógica para detecção do sensor de toque e do sensor de toque ultrassônico

Observe, na imagem a seguir, como ficou o programa completo, a parte posterior à sequência de blocos lógicos que incrementamos é praticamente a mesma. Agora, o Robô Explorer é capaz de livrar-se de obstáculos detectados pelo sensor de toque.

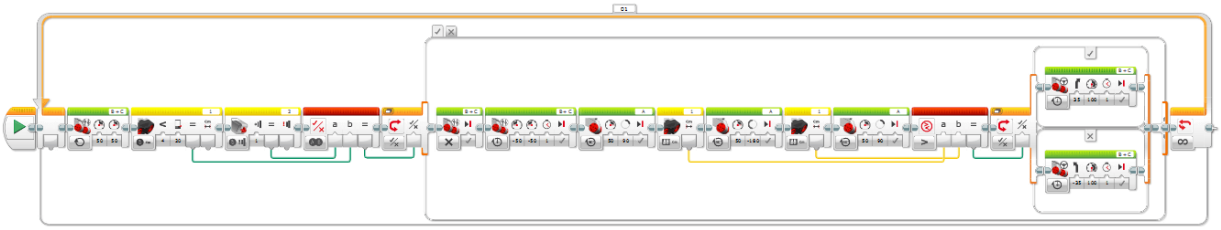


Figura 21.11: Robô Explorer - Variação 1: programa completo

Como se pode observar, neste código não foi utilizado o bloco `Wait` , no lugar dele foi adicionado o bloco `Move Tank` no modo `Off` . O bloco `Move Tank` não exerce a função de fazer o robô esperar, porém, até os motores encerrarem suas atividades e o movimento seguinte seja efetuado, ocorre um pequeno intervalo entre as ações.

**2.**

Para esta variação, vamos adicionar código da variação anterior mais alguns complementos. O programa anterior será aproveitado, pois assim não é necessário criar todo o programa novamente. Adicionaremos o seguinte complemento ao código: caso o sensor de toque permaneça pressionado, o veículo dará marcha à ré por alguns instantes para se afastar do objeto. Após efetuar o movimento de ré, o Explorer prosseguirá com suas ações.

Os códigos feitos anteriormente também continham o movimento de marcha à ré. Porém, o movimento era realizado sempre que o programa entrava no `Switch` , independente se a detecção havia sido feita pelo sensor ultrassônico, ou pelo sensor de toque. Apenas obstáculos detectados pelo sensor de toque obstruem o movimento do robô, pois o sensor ultrassônico realiza a detecção assim que os obstáculos fiquem a uma distância menor que 20 cm. Dessa forma, o movimento de marcha à ré torna-se necessário e realmente eficaz quando aplicado as detecções realizadas pelo sensor de toque. Logo, com está variação o robô ficará mais inteligente.

Para que o robô execute o movimento de marcha à ré, vamos inserir no `Switch – Logic` mais um `Switch` . Este, por sua vez, se trata de um

`Switch` configurado no modo `Touch Sensor – Compare – State` . Dentro desse `Switch` , inseriremos no `Case True` o bloco responsável pela ré do robô. O `Case True` significa que a condição comparada obteve um valor `True` , ou seja, o sensor de toque está pressionado. Assim, quando o sensor de toque estiver pressionado, o `Switch` entra em ação e executa o movimento de marcha à ré.

Para o movimento de marcha à ré, neste código, utilizamos o bloco `Move Tank` no modo `On for Seconds` , ajustado para um segundo, que é suficiente para que o robô escape do obstáculo. Como o movimento é de ré, ou seja, em sentido oposto, utilizamos valores negativos para que o movimento ocorra.

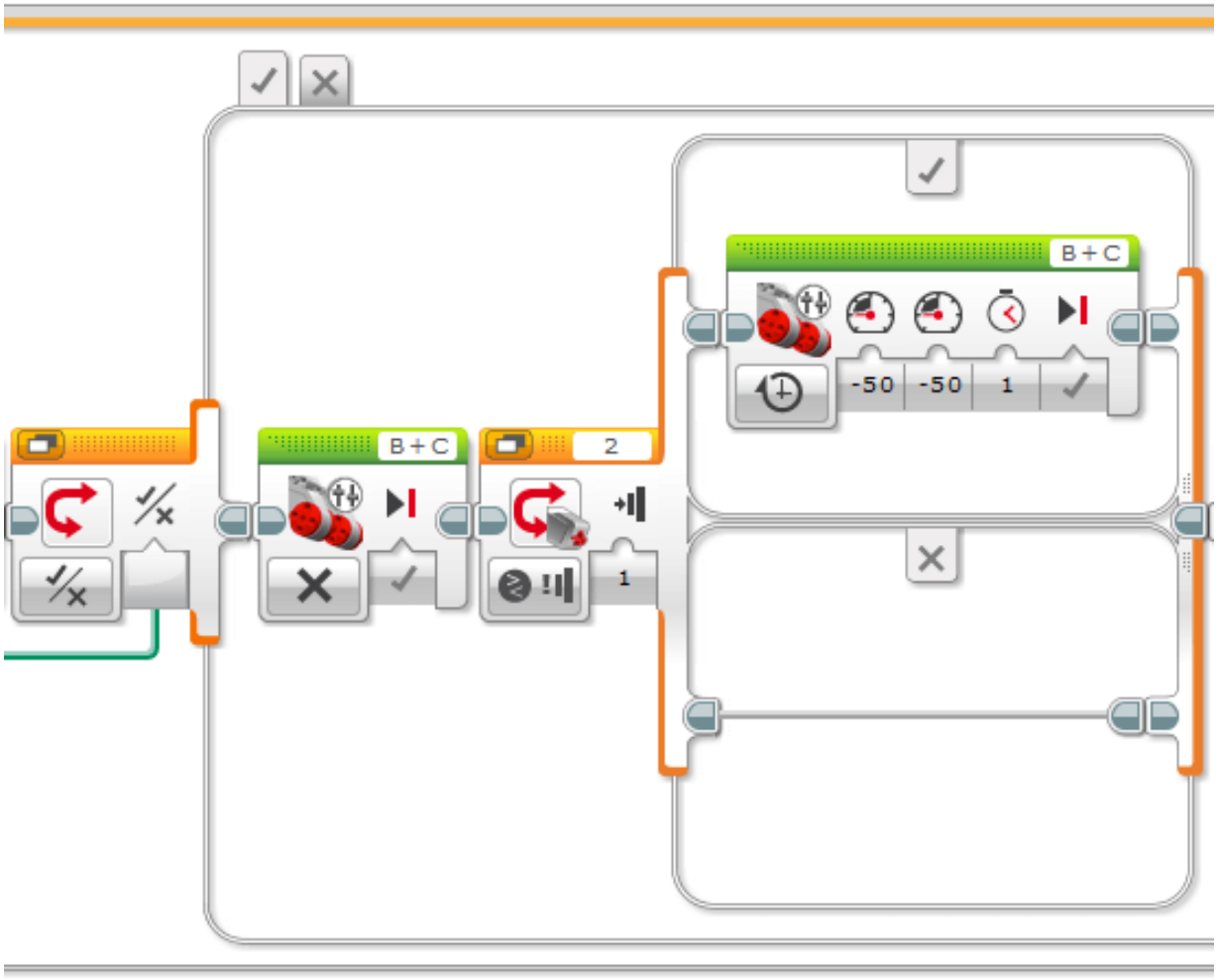



Figura 21.12: Robô Explorer - Variação 2: Switch - Touch Sensor contendo o bloco responsável pela marcha à ré do robô

Na imagem a seguir você pode visualizar como ficou o código completo desta variação.

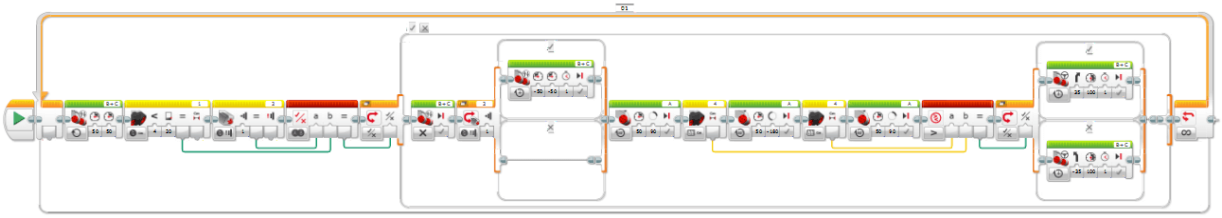


Figura 21.13: Robô Explorer - Variação 2: Programa Completo

## 21.2 Exercícios

**1.**

Considere a variação número 1, feita anteriormente. Acrescente a seguinte restrição ao programa: após o protótipo detectar três vezes a proximidade de objetos, seja pelo sensor ultrassônico ou via colisão por sensor de toque, o programa será desativado.

**2.**

Altere o programa anterior para desativar o robô após seus sensores detectarem três vezes a proximidade de objetos ou colidirem com algo. O protótipo deve parar por 10 segundos, quando então repetirá indefinidamente o ciclo descrito.

## 21.3 Resoluções

**1.**

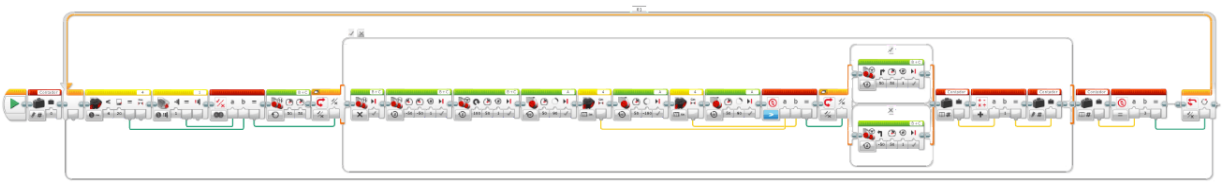


Figura 21.14: Solução da questão 1

Para que você não precise reescrever todo o programa de novo, você pode utilizar o código da variação 1 como base e, a partir dele, ir desenvolvendo as alterações propostas no enunciado da questão. Portanto, as imagens a seguir (antes da explicação detalhada do programa) correspondem aos incrementos e alterações realizadas no código da variação 1.

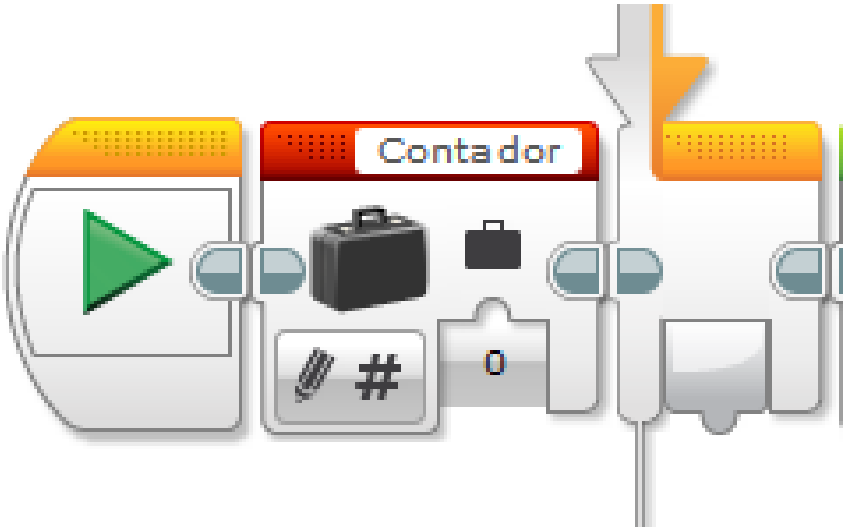



Figura 21.15: Solução da questão 1 - Detalhe 1

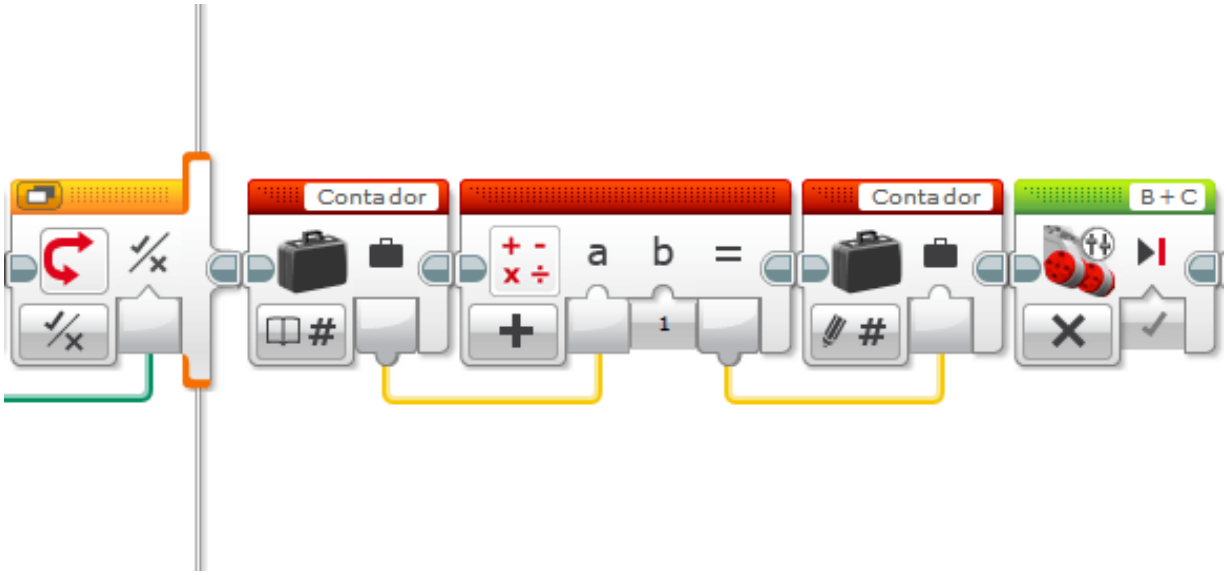



Figura 21.16: Solução da questão 1 - Detalhe 2

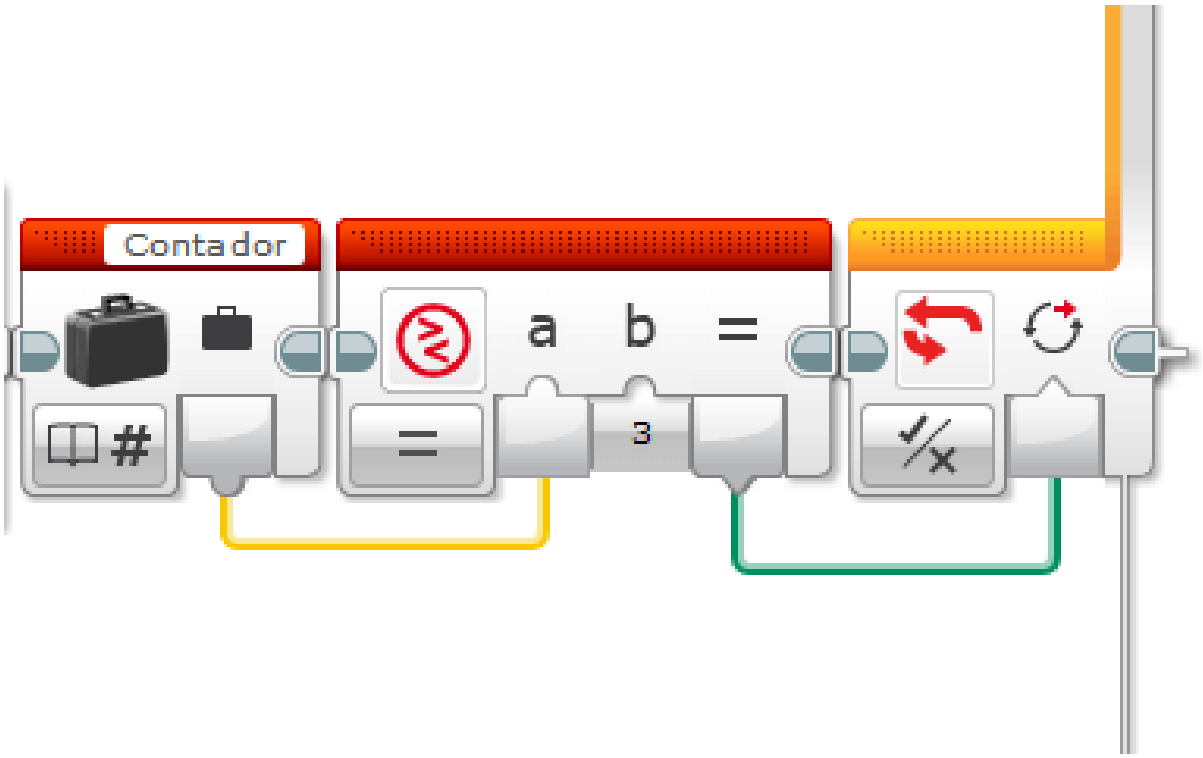



Figura 21.17: Solução da questão 1 - Detalhe 3

Para este programa serão utilizadas algumas variáveis. A variável responsável por armazenar o número de detecções será chamada de `Contador` . Ela deve ser colocada fora do `Loop` , e no modo `Write -`

`Numeric` , com o valor `0` , pois o programa não começou a ser executado ainda e não fez nenhuma detecção. Com isso, toda vez que o programa for iniciado a variável `Contador` será iniciada com valor `0` .

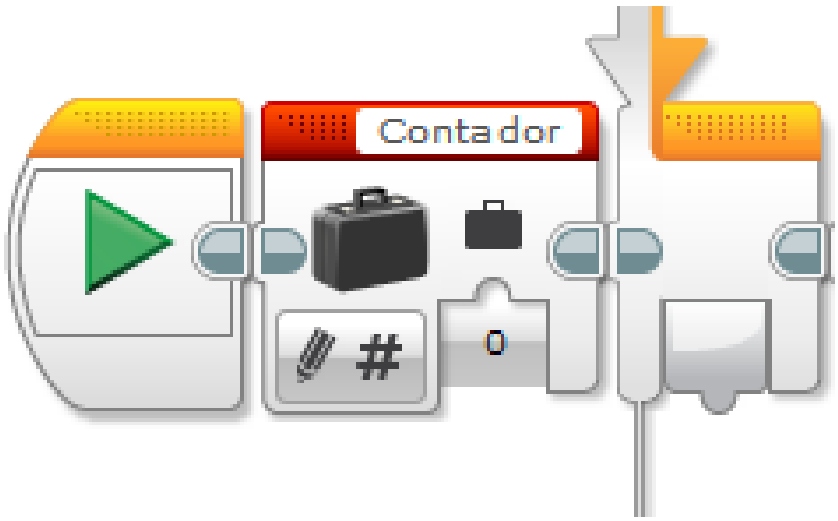



Figura 21.18: Solução da questão 1 - Variável “Contador”

Agora temos a parte responsável pela contagem e armazenamento do número de detecções. Para isso, foi utilizada a variável `Contador` no modo leitura ( `Read - Numeric` ) e o bloco de dados `Math` no modo `Add` . O valor contido na variável `Contador` é enviado, por meio do `Data Wires` , para a entrada `a` do bloco `Compare` . Na entrada `b` escreve-se o valor `1` . Em seguida, é só adicionar outra vez a variável `Contador` , dessa vez, no modo `Write - Numeric` . O resultado da operação efetuada pelo bloco `Compare` deve ser registrado na variável `Contador` . Para isso, basta conectar enviar o resultado contido na saída `Result` do bloco `Compare` para a entrada `Value` da variável `Contador` . Agora, sempre que o programa fizer uma detecção, esse valor será registrado na variável.

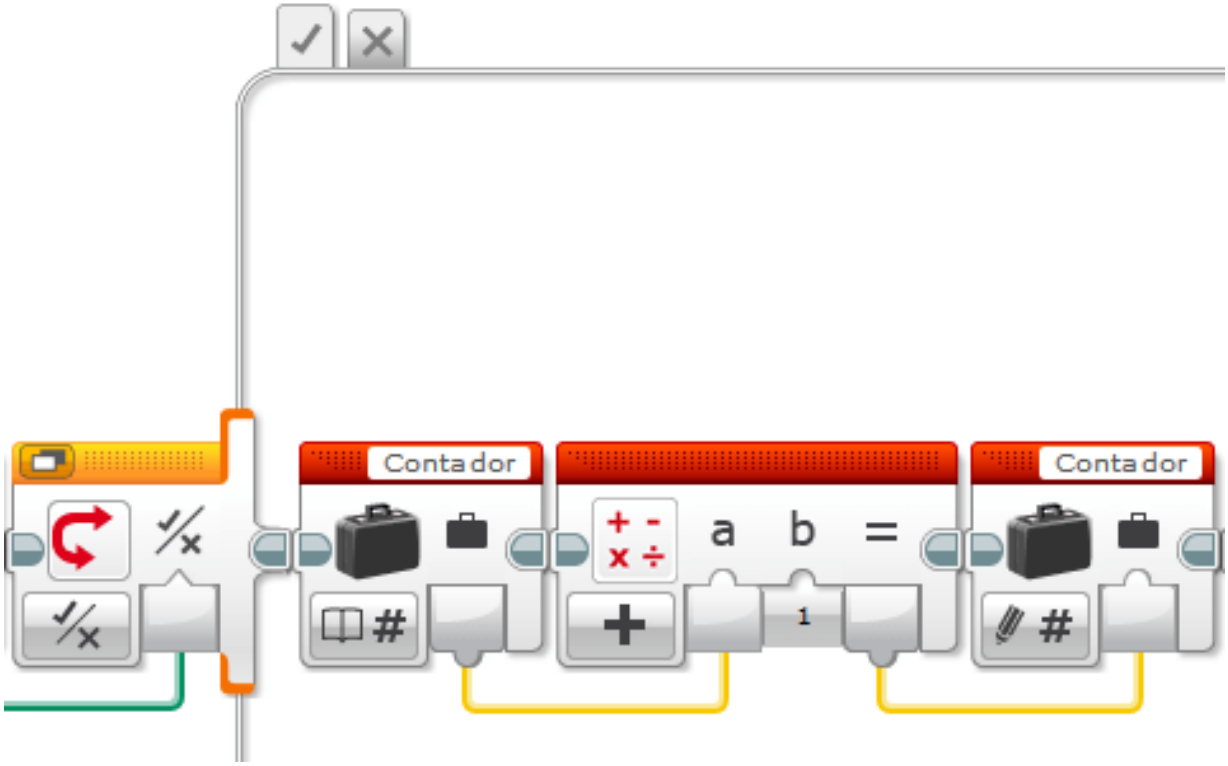



Figura 21.19: Solução da questão 1 - Blocos responsáveis pelo incremento

Após a sequência de blocos responsáveis pelo incremento, temos a parte do programa que realiza as funções de: parar, efetuar marcha à ré, medir a distância de cada lado, realizar o giro e, então, encerra-se o `Switch` . Até aqui, nada de novo e, o `Loop` ainda permanece no modo `Unlimited` .

Com a sequência de blocos responsável pelo incremento pronta, agora, é necessário criar a parte do código responsável pela leitura do valor armazenado na variável `Contador` , e que encerre o programa quando o valor da variável for igual a `3` . O primeiro passo é configurar o `Loop` para o modo `Logic` . Por meio do bloco de dados `Compare` , configurado no modo `Equal To` , fornece-se o resultado lógico ao `Loop` , a partir do qual tomará a decisão de parar a execução do programa ou não. Ou seja, quando o valor da variável `Contador` for igual a `3` ele envia um valor lógico `True` que faz com que o `Loop` encerre suas atividades, mas, enquanto o valor da variável for

menor que três, o valor da comparação será `False` , logo, o programa continua a ser executado.

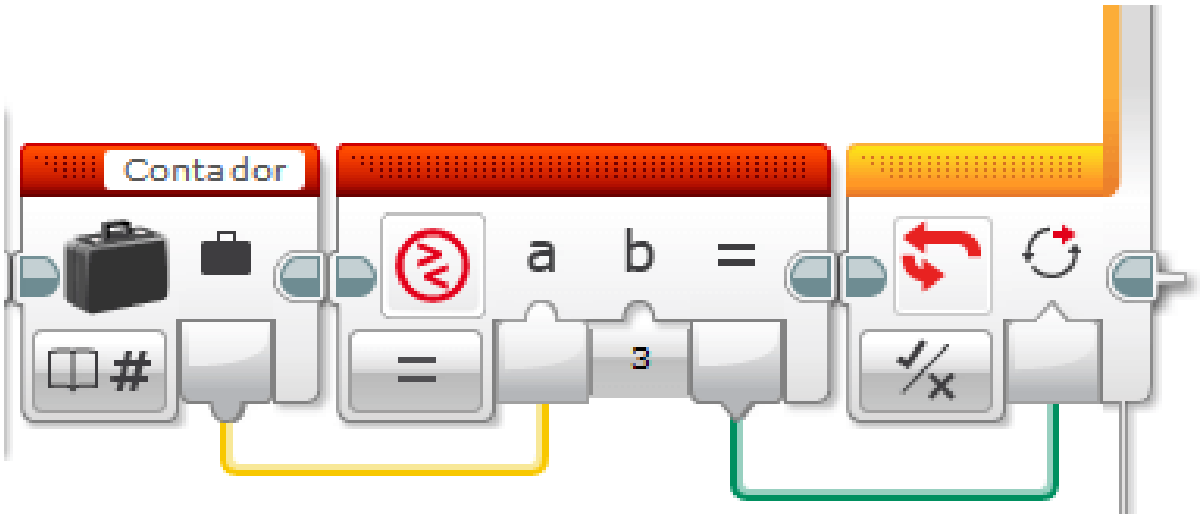



Figura 21.20: Solução da questão 1 - Sequência de blocos para desativar o robô após 3 detecções

**2.**

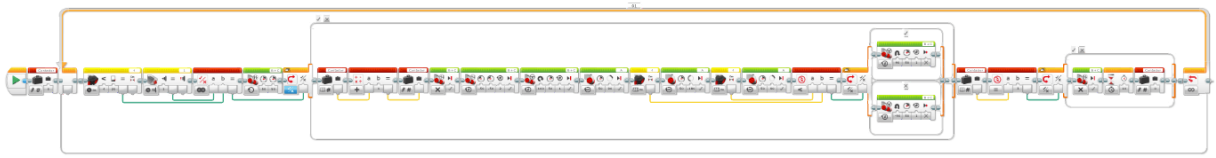


Figura 21.21: Solução da questão 2

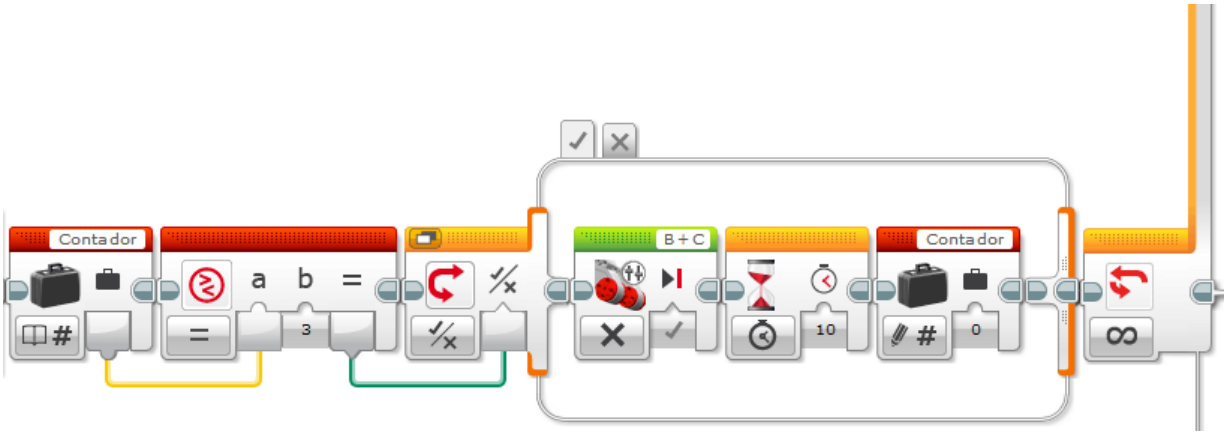



Figura 21.22: Solução da questão 2 - Detalhe - Modificação realizada na parte final do código

Para este exercício serão utilizados conceitos anteriormente abordados nas variações do Robô Explorer. Para resolver este exercício, você pode usar o código da atividade 1 como base.

A primeira modificação deste código será no `Loop` , pois o robô vai executar o programa indefinidamente, ele só vai parar por 10 segundos. Por isso, o `Loop` deve ser configurado no modo `Unlimited` . Mas todos os blocos anteriores permanecem. Nenhuma modificação a mais precisa ser feita no código.

No exercício anterior, o resultado da comparação era enviado à entrada `Until True` do `Loop` , configurado no modo `Logic` . Porém, desta vez, esse resultado será enviado para um `Switch - Logic` . Nesse `Switch` será programada a parte do código responsável por parar o robô por 10 segundos após fazer as 3 detecções. Portanto, quando a variável `Contador` tiver registrado 3 detecções, ela envia um resultado lógico `True` para o `Switch` , que entra em ação.

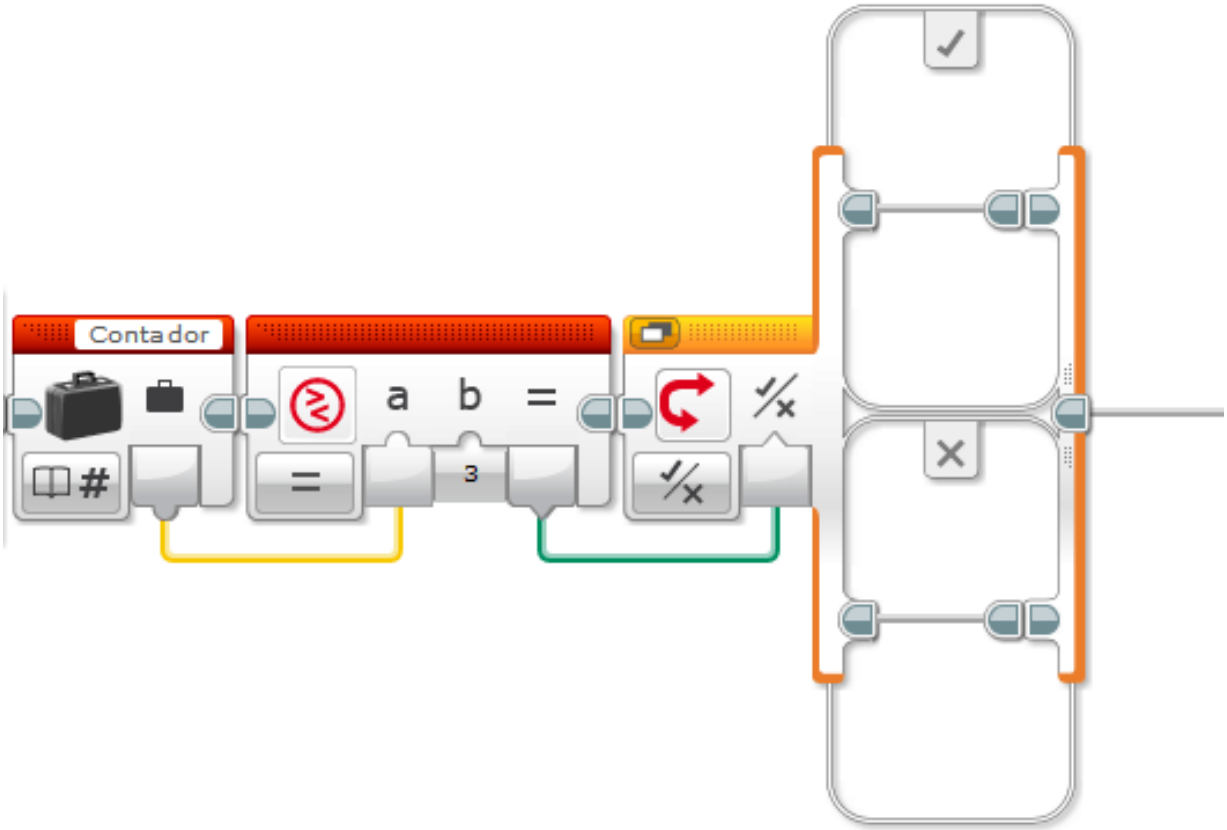



Figura 21.23: Solução da questão 2

No `Switch` , adicionamos o bloco `Move Tank` no modo `Off` , desta forma, quando o robô tiver registrado 3 detecções, ele desliga seus motores. Logo após, vem o bloco `Wait` , configurado no modo `Time Indicator` e ajustado parar esperar por 10 segundos. Por último, inicializa-se a variável `Contador` novamente, atribuindo a ela o valor numérico zero. A variável deve ser configurada no modo `Write` com o valor `0` , pois, após parar, o robô deve zerar a contagem para depois iniciá-la novamente. Como o `Loop` está no modo `Unlimited` todas essas ações serão executadas infinitamente, sempre que o robô realizar as três detecções ele para e espera para prosseguir novamente.

Figura 21.24: Solução da questão 2

Veja como ficou a parte final do programa:

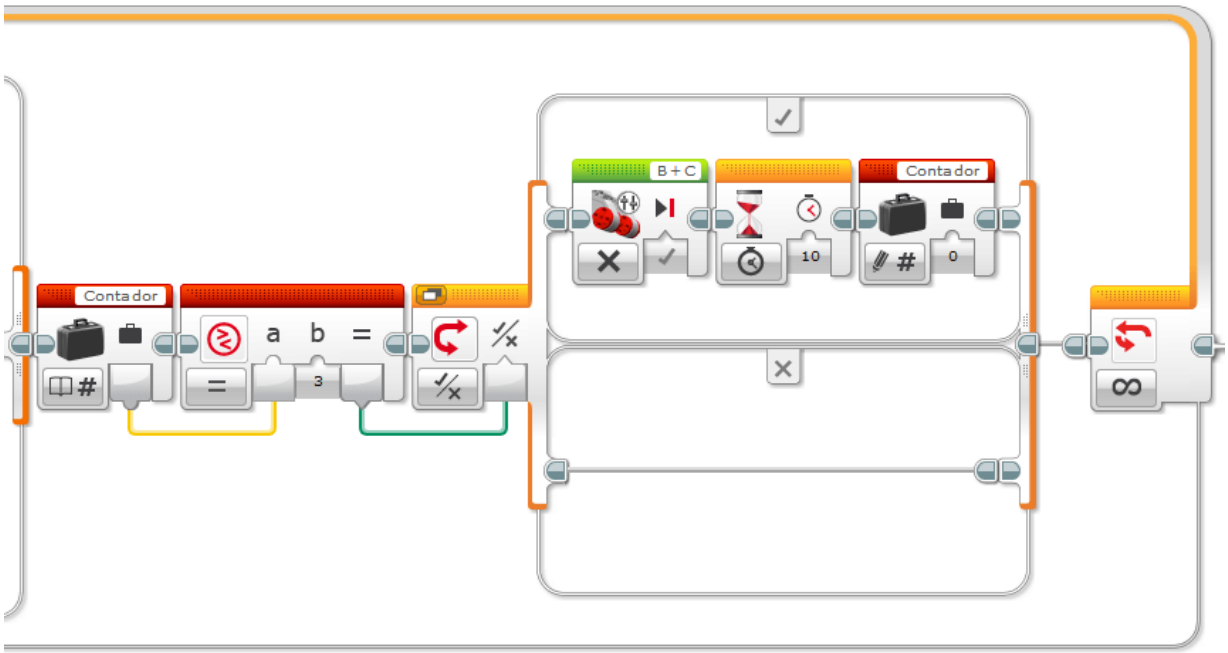



Figura 21.25: Solução da questão 2

## Pronto! E agora?

Aqui se encerra este capítulo, esta foi nossa última experiência. Esta foi a que apresentou um nível de maior complexidade em relação às outras. A cada variação do código, novas funções e etapas foram incluídas, proporcionando ao robô novas funcionalidades. Nesta experiência você utilizou vários blocos que

foram trabalhados durante o livro e pôde utilizar boa parte de seu conhecimento para a construção de um programa mais completo e desafiador.

# Capítulo 22
# Conclusão

Chegamos ao fim de mais uma parte do livro. Trabalhamos com três protótipos de robôs e desenvolvemos vários programas. Por meio destes três protótipos, você pôde aprender e aperfeiçoar suas habilidades em programação. Ao desenvolvermos os programas de cada protótipo vimos muitos blocos novos. Por meio das propostas dos enunciados, fomos conhecendo e aprendendo sobre os blocos que nos ajudam para cada situação.

A partir dos objetivos definidos no enunciado criamos programas e fomos ainda mais além. Em cada capítulo foram trabalhadas variações, mostrando formas e lógicas alternativas para solucionar os problemas, e acrescentando ainda mais funcionalidades aos programas. Você também contou com exercícios, por meio dos quais pôde testar seus conhecimentos.

Mas o objetivo principal dos exercícios foi estimular seu raciocínio e fazer com que você desenvolvesse suas habilidades ainda mais. Por isso, todos os exercícios contaram com repostas ao fim do capítulo, ensinando quais os passos necessários para a resolução dos problemas.

A próxima parte do livro ainda contém mais um pouco de programação. Nela serão apresentadas algumas formas alternativas dos programas que trabalhamos e você conhecerá mais alguns blocos. Além disso, na parte seguinte, você encontrará informações adicionais e dicas que contribuirão ainda mais para o seu aprendizado e experiência com o kit EV3.

# RECURSOS AVANÇADOS

Esta parte do livro, além de dar algumas dicas relacionadas à programação, também ensina e mostra alguns recursos um pouco mais avançados que podem ser utilizados para aperfeiçoar o seu programa e suas habilidades. Aqui você encontrará dicas e boas práticas de programação, como calibrar o sensor de cor, organização de tarefas e divisão de tarefas. Esses recursos são importantíssimos para seu currículo, pois, a partir deles, você aprenderá como resolver problemas mais complexos sem muito esforço.

Nesta parte do livro, também serão feitos mais alguns programas. Serão abordadas formas alternativas e de melhor resultado para resolver alguns dos problemas propostos nos capítulos anteriores.

**E a dica/curiosidade n°1 é:**

Se você estiver com seu Brick ligado e conectado à sua máquina é possível fazer download e executar seu programa com apenas um clique. Para fazer isso, basta clicar no bloco `Start` do seu programa!

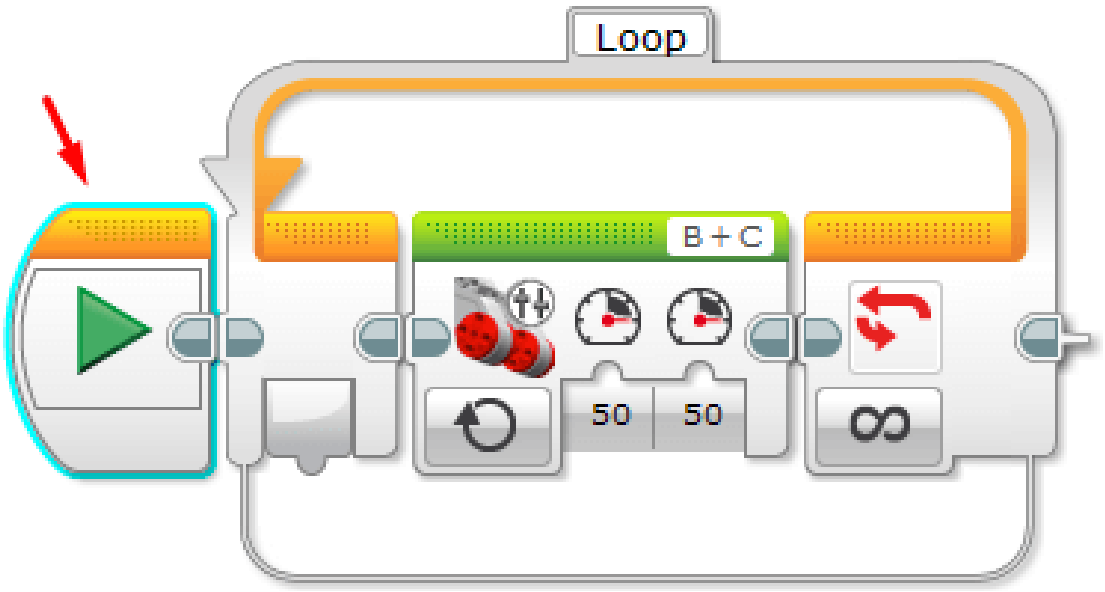



Figura 1: Bloco Start

# Capítulo 23
# Boas práticas de programação no EV3

Boas práticas são fundamentais na programação, até mesmo em uma linguagem gráfica, como é o nosso caso. É importante ter cuidados na “escrita” do programa. Um programa bem escrito é aquele que possibilita leitura clara e fácil compreensão, tanto ao seu desenvolvedor, quanto a outro programador que tenha acesso a ele. Um programa bem feito faz toda a diferença. Nós, enquanto desenvolvedores, devemos produzir um código eficiente, direto e simples.

## 23.1 Análises e testes

Também é muito importante realizar testes, pois antes de tudo o programa deve funcionar. Por isso, sempre teste o seu código, observando se todas as suas etapas estão sendo cumpridas e executadas corretamente.

Somente testando é que nós podemos verificar com 100% de certeza se o programa realiza todas as funções esperadas. Mesmo que em tese nosso código esteja correto, o ambiente e suas adversidades devem ser levados em conta. Por exemplo, ao criar um robô seguidor de linha, o robô funciona a partir da leitura da luminosidade refletida, e isso é uma variável. A quantidade de luminosidade refletida está diretamente associada ao ambiente, pois alguns materiais refletem melhor e outros não. Por isso, é altamente recomendável e importante que você sempre teste seus programas.

A partir de testes e observações é possível aperfeiçoar o programa. Muitas vezes haverá uma interação com o ambiente externo, quando se utilizam sensores, por exemplo. Estes sensores coletam dados do ambiente, e quando é realizada a análise dos dados pode-se garantir um melhor funcionamento do robô. E conforme vimos, isso é observado no robô seguidor de linha.

## 23.2 Comentários

A primeira coisa que podemos fazer no código é adicionar o bloco `Comment` , que fica na aba *Advanced*. Ao inseri-lo no programa, basta clicar nele e escrever o nosso comentário.

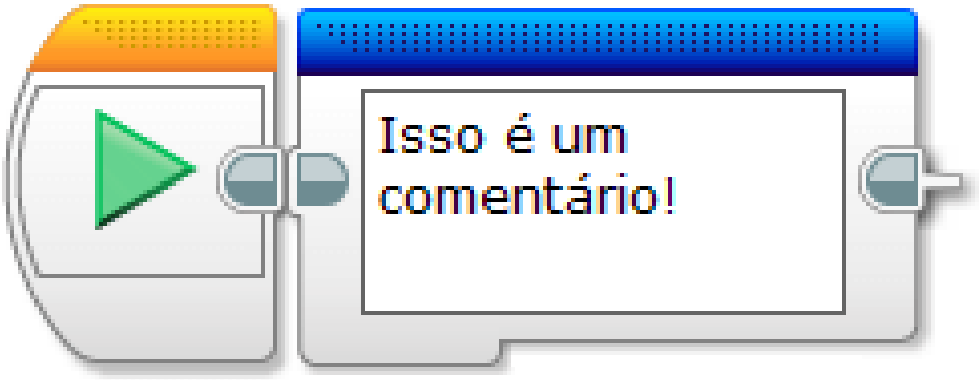



Figura 23.1: Inserindo um comentário com Bloco "Comment"

Além do bloco `Comment` que pode ser inserido no programa, também podemos adicionar blocos de comentário ao entorno do programa, detalhando ainda mais cada passo.

Vamos ver como fazer isso:

Passo 1: clique em `Comment` , que está situado no canto superior direito.

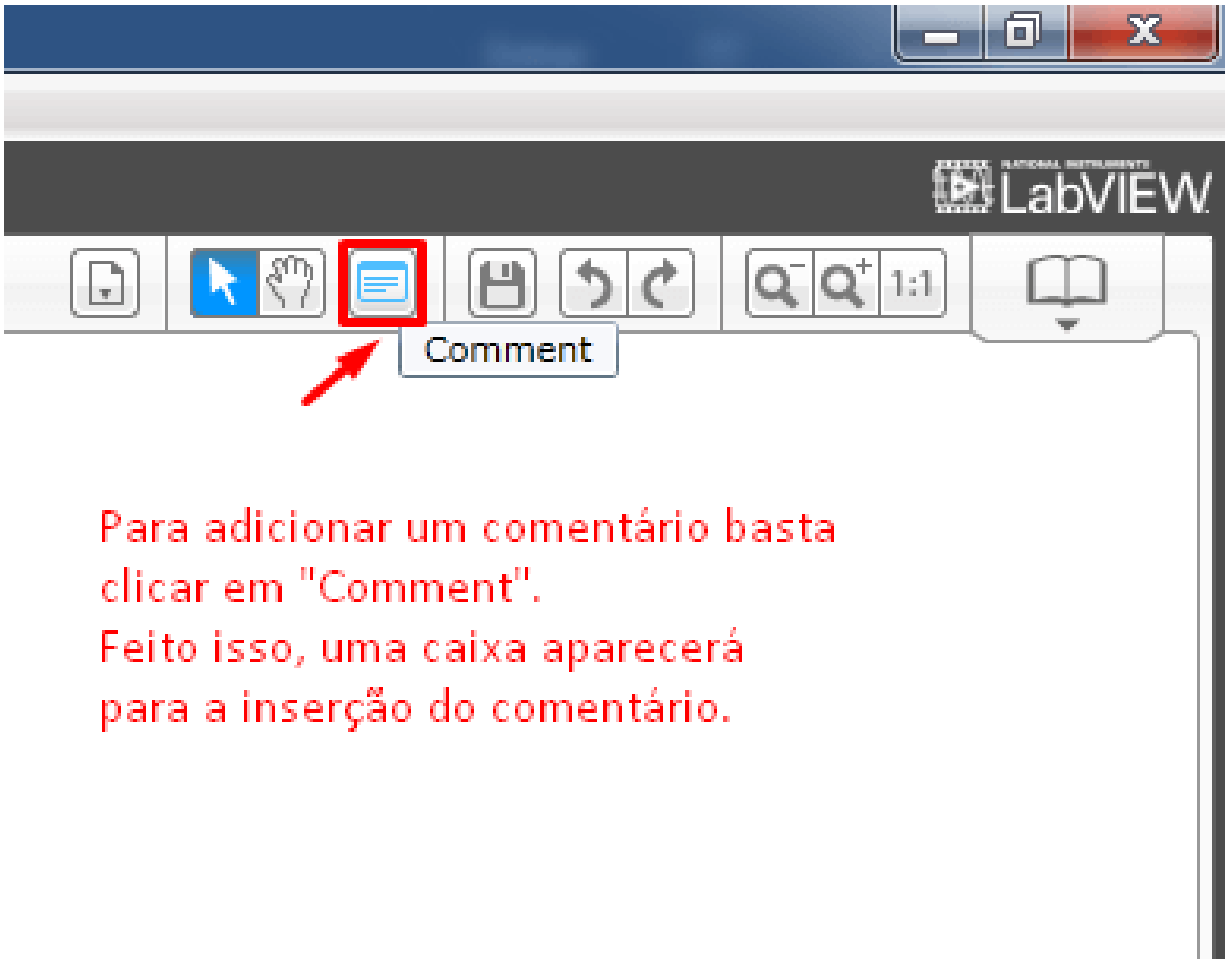



Figura 23.2: Inserindo um comentário

Passo 2: adicione o seu comentário na caixa de textos.

Figura 23.3: Inserindo um comentário

Outro fator que melhora muito o entendimento e organização do nosso programa é nomear suas estruturas de controle. Podemos adicionar nomes às estruturas de controle clicando na pequena caixa situada acima das estruturas. É sempre bom dar nomes sugestivos à função das estruturas de controle, por exemplo, indicando se trata-se de um "Loop Interno" ou "Loop Externo", ou numerando as estruturas de controle. O importante é que você mantenha sempre uma boa organização do seu código, de maneira que você entenda cada parte do código, pois esse é um fator que ajuda muito em uma análise e resolução de um possível erro apresentado pelo programa.

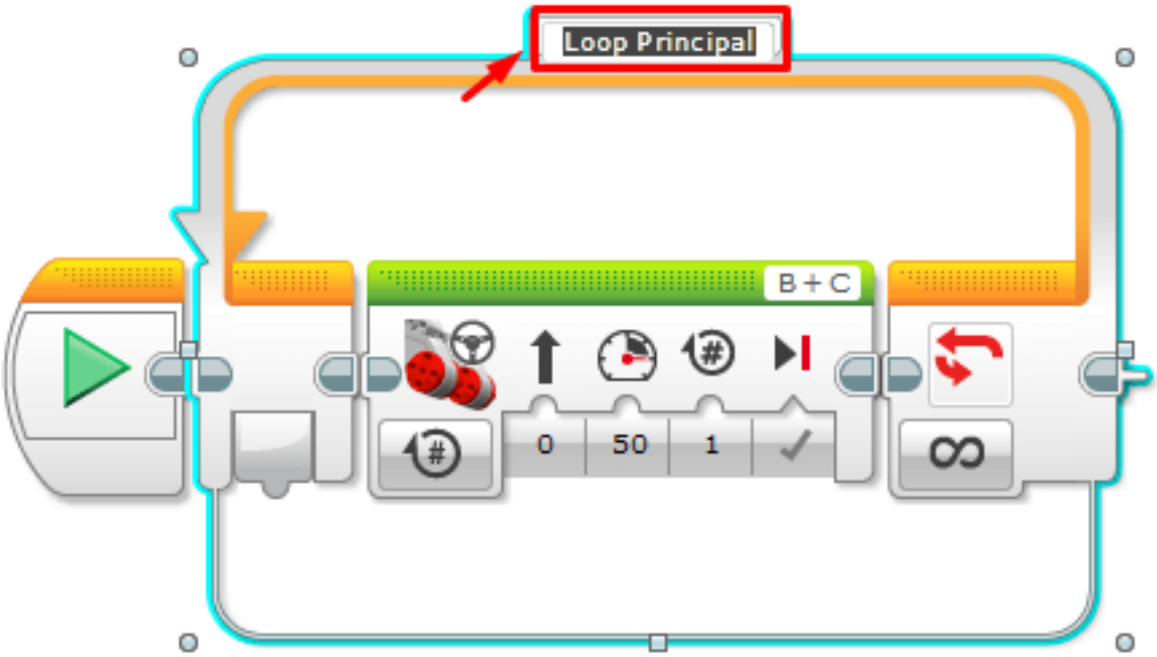



Figura 23.4: Nomeando as estruturas de controle

Para enriquecer o nosso projeto, é possível adicionar ainda fotos, vídeos e descrição. Isso é muito útil quando se tem a intenção de compartilhar o projeto, pois além de explicar e descrever o código, você pode adicionar imagens e vídeos exibindo o funcionamento do programa. Vimos isso logo nas primeiras páginas deste livro.

Outra dica é dar nomes bastante sugestivos às variáveis, isso torna o programa mais organizado e facilita até mesmo sua construção, pois desta forma você (e quem mais estiver analisando o programa) não corre o risco de confundir as variáveis usadas.

Essas dicas aqui apresentadas têm o intuito de facilitar e melhorar seu processo de aprendizagem. Com elas, você poderá melhorar sua organização e escrita do programa, tornando-o acessível a todos.

# Capítulo 24
# Organização das tarefas

Você deve ter reparado que, quando fizemos o programa *Detector de presen  a*, a sequência de blocos do nosso programa não estava disposta de maneira totalmente horizontal.

A forma mais comum de “escrever” o programa é dispor os blocos um após o outro, na mesma direção, ou seja, formando uma sequência horizontal. Observe a figura a seguir.

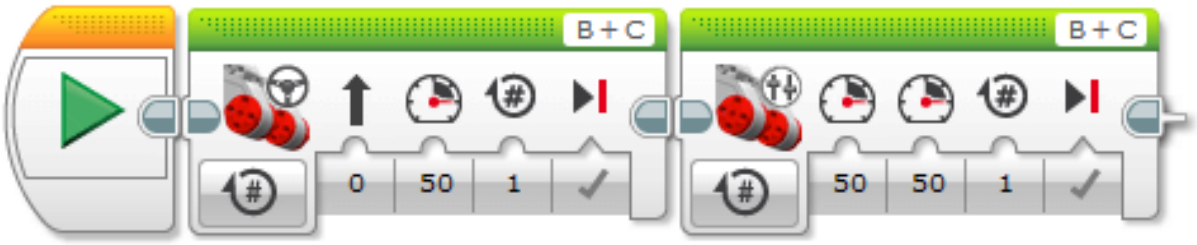



Figura 24.1: Modo Horizontal

Porém, podemos possível organizar os blocos de outra maneira, aproveitando o espaço vertical, como no exemplo a seguir:

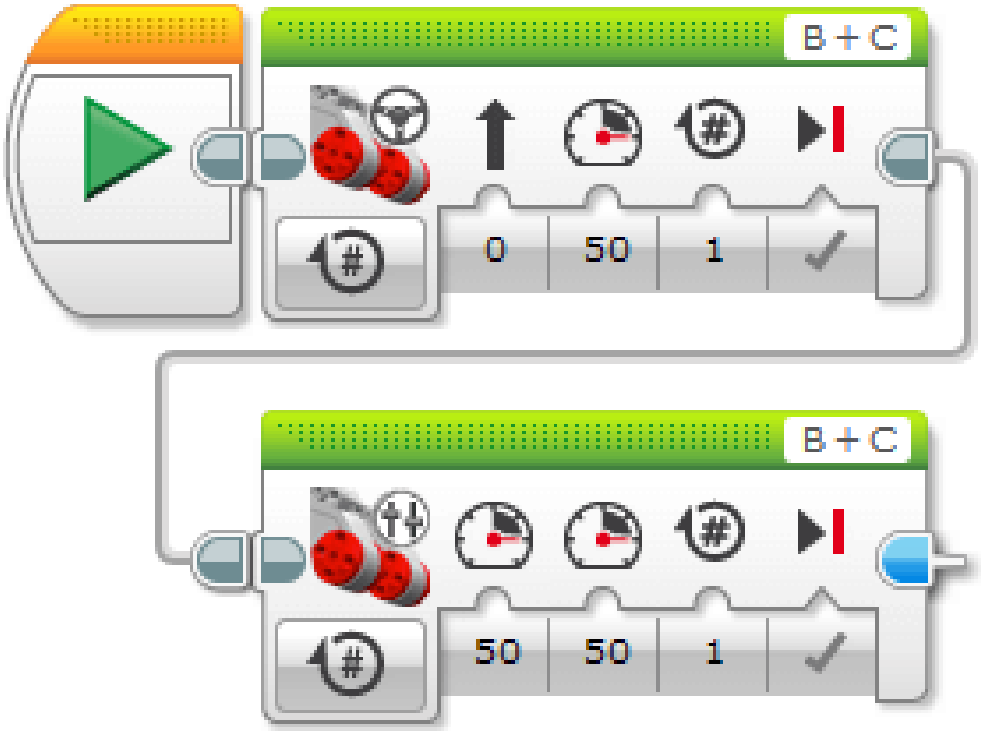



Figura 24.2: Modo Vertical

Neste caso, precisamos ter cuidado quanto à ligação dos blocos (feita pelo fio cinza), pois isso afeta o funcionamento do programa. Essa outra forma de arranjo dos blocos pode ser útil quando se tem um programa muito extenso por facilitar a navegação, já que é possível concentrar mais blocos, aproveitando todo o espaço da tela.

Para desfazermos o arranjo e voltarmos para a organização dos blocos na horizontal, basta clicarmos na ponta de onde sai o fio que realiza a conexão.

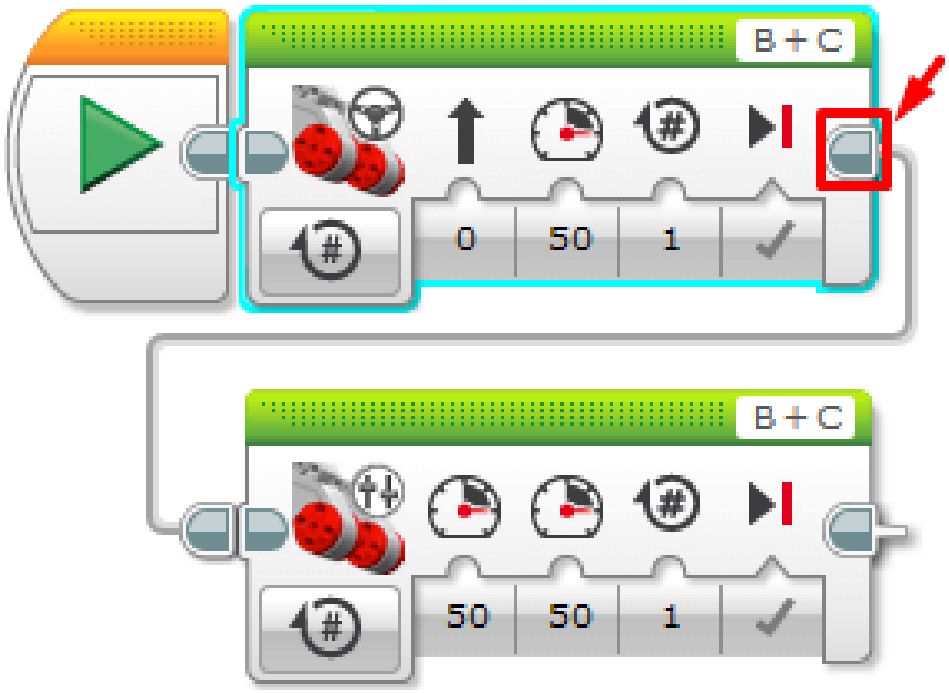



Figura 24.3: Modo Vertical - Voltando ao modo Horizontal

### Observação

É preciso tomar muito cuidado com as conexões entre eles: os blocos devem ser ligados uns aos outros.

Por exemplo, no código a seguir, perceba que o segundo bloco `Move Tank` (de cima para baixo) não está conectado aos outros, e por isso ele apresenta uma cor mais clara. Olhando pelo fio de dados isto parece bem óbvio, porém, um erro desses pode muito bem passar desapercebido.

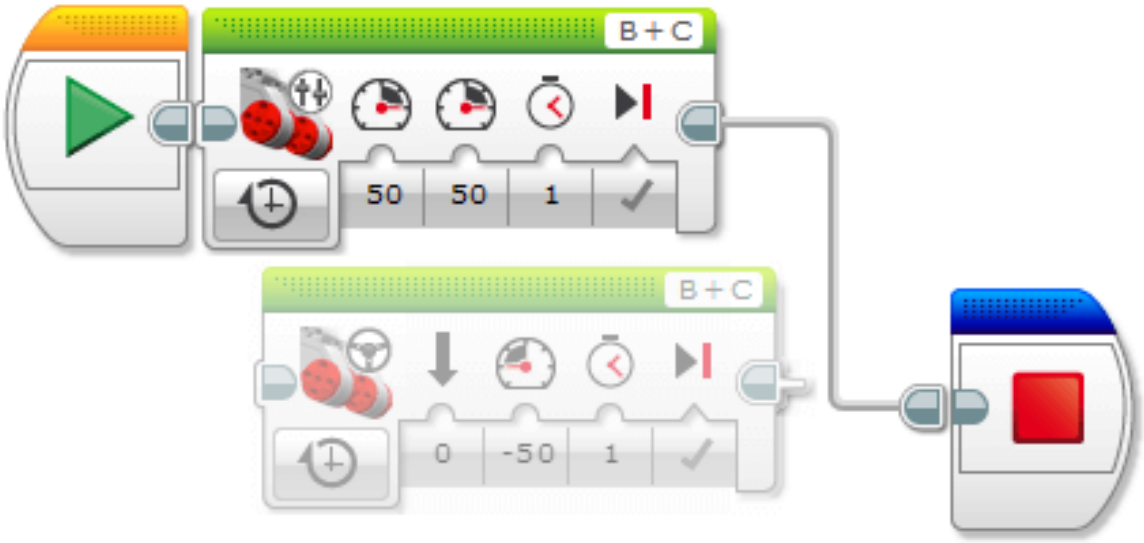



Figura 24.4: Exemplo de blocos desconectados

Agora você conhece mais uma forma de melhorar a organização do seu programa!

# Capítulo 25
# Divisão de tarefas

O software EV3 permite a divisão de tarefas (assim como seu antecessor, o NXT). Este recurso consiste na divisão do fluxo do programa. Podemos utilizá-lo para executar simultaneamente duas partes distintas do programa, sem que uma dependa da outra.

Realizando esta divisão, o programa tem dois “caminhos”, que são executados ao mesmo tempo, como se fosse uma espécie de `Switch` improvisado. Porém, é preciso ter cuidado ao usar esse recurso, já que isso pode gerar muitos problemas se for usado de maneira incorreta. Por exemplo, deve-se sempre verificar cuidadosamente se todos os fios estão conectados de maneira correta, pois pode haver um ou outro bloco que fique desconectado, no caso de alguma desatenção ao realizar as ligações durante o processo (caso esteja desconectado, ele ficará em uma cor meio transparente).

Exemplificando:

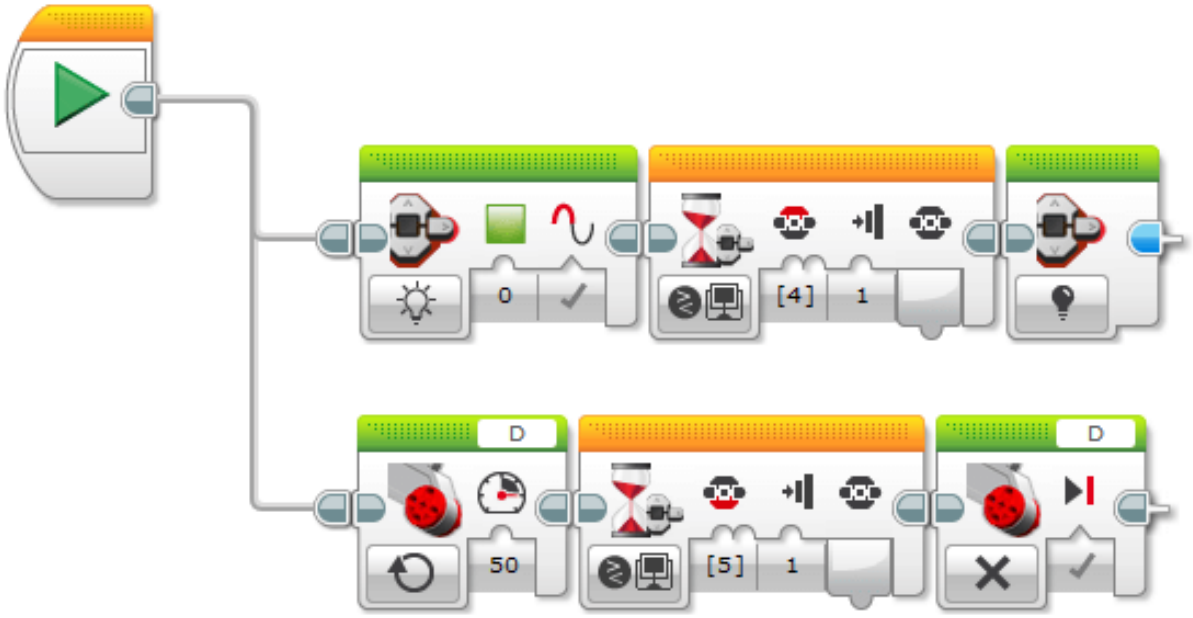



Figura 25.1: Divisão de tarefas

Com a divisão do fio ( `Data wires` ) vindo do bloco `Start` , esse programa executa os blocos da parte superior e inferior simultaneamente.

Ele acende o `Brick Status Light` , que permanece aceso até que o botão superior do Brick seja pressionado. Isso é feito por meio do bloco `Wait` , que aguarda até que a condição seja cumprida.

Quando identificado o acionamento do botão, a próxima ação será o bloco `Brick Status Light` no modo `Off` , ou seja, desligado. Mas, enquanto isso, a parte inferior do programa continua a ser executada, proporcionando a movimentação do motor. O motor só entra no modo `Off` (é desligado) quando detectar que o botão inferior do Brick foi desligado. O programa só finaliza com o encerramento das duas tarefas.

### Utilizando o bloco `Start`

Outra forma de se obter o mesmo resultado consiste em utilizar dois blocos `Start` , um para cada tarefa.

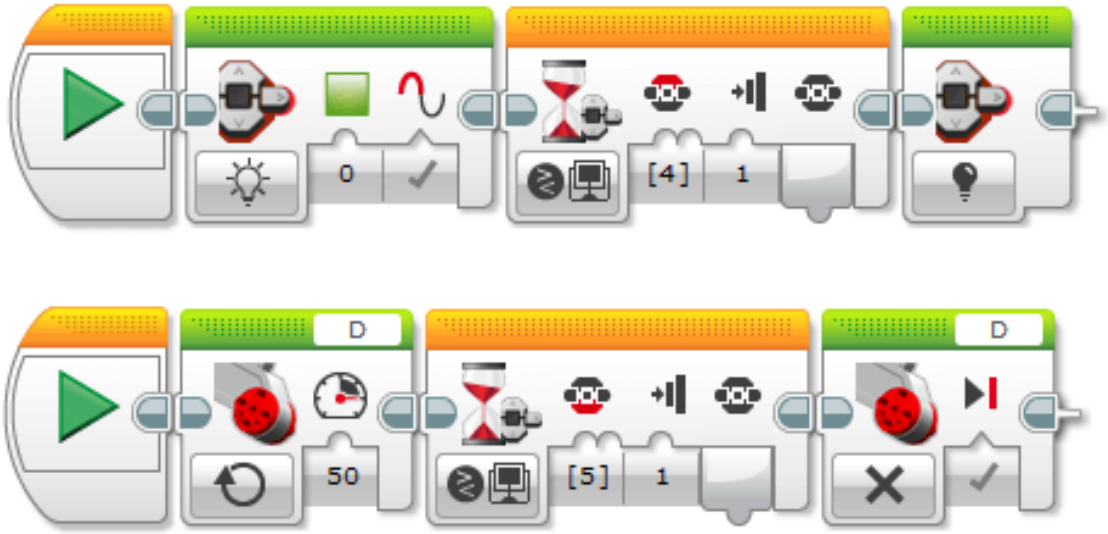



Figura 25.2: Divisão de tarefas com Bloco Start

Aqui foram apresentadas algumas possibilidades que você poderá utilizar em seus programas futuros.

# Capítulo 26
# Calibrando o Colour Sensor

Um sensor bem calibrado proporciona uma enorme diferença no desempenho do nosso programa. Com sensores bem calibrados, nosso robô executa ações de maneira mais precisa. Dentre os fatores que atrapalham a leitura do sensor de cor no modo de intensidade da luz refletida, por exemplo, pode-se destacar a própria luz ambiente.

Com uma rotina de calibração é possível ajustar valores máximos e mínimos do sensor. Para um melhor desempenho do programa (que utilize o sensor cor no modo de intensidade de luz refletida), é recomendado que sempre se execute a rotina de calibração. Dessa forma, tem-se valores mais precisos, evitando discrepâncias na leitura da superfície na qual o sensor está apontado.

## Criando uma rotina de calibração

Como já foi explicado, o bloco `Colour Sensor` oferece o modo `Calibrate`. Nesse modo, é possível calibrar o valor mínimo, máximo e ainda reiniciar o sensor para que ele volte às configurações iniciais.

A seguir está o programa:

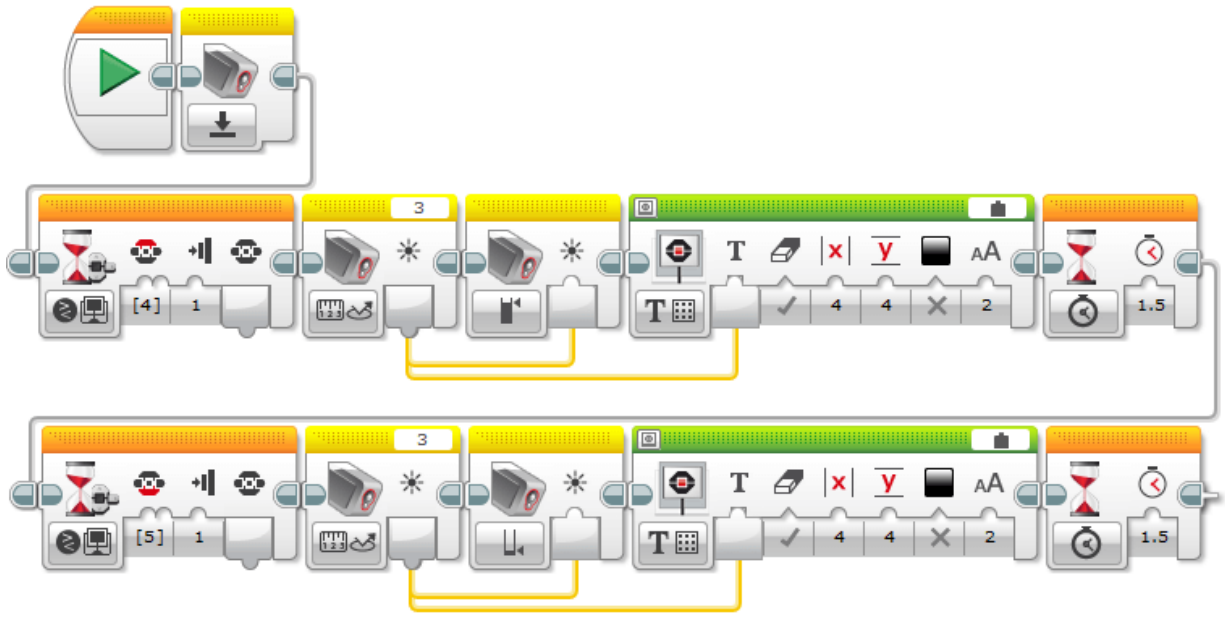


Figura 26.1: Rotina de calibração

A essa altura do livro você já deve ter conseguido analisar e interpretar o funcionamento desse programa. Mas, para que nenhuma dúvida perdure, vamos entendê-lo bloco a bloco.

Se você quiser acessar a rotina de calibração no seu navegador, ela está disponível no seguinte link:

https://github.com/vitor000x/ev3_Codes/blob/master/Recursos-Avancados/Rotina-de-Calibracao/RC.ev3

## Analisando o código

O primeiro bloco do programa é o bloco `Colour Sensor` no modo `Reset`, com isso o sensor será reiniciado e, então, pode-se calibrar o sensor. O programa funciona da seguinte maneira: para efetuar a calibração é necessário que se apertem os botões do Brick.

A primeira calibração ocorre para o valor máximo. Para isso, é só apertar o botão `Up` do Brick. Após feito isso, o sensor medirá o valor, registrará como valor máximo, exibirá esse valor na tela e esperará por 1.5 segundos. A próxima etapa é definir o valor mínimo. O

mesmo processo de antes é feito, porém, neste caso, você precisa apertar o botão `Down` .

> **Observação**
>
> Antes de realizar a calibração, certifique-se de que o sensor está posicionado no local desejado e, se estiver, faça a calibração.

Agora, sempre que você utilizar seu robô seguidor de linha (ou algum outro programa com lógica semelhante) terá à sua disposição uma rotina de calibração. Dessa forma, você garantirá melhores resultados para o seu robô.

Com os sensores bem calibrados você não terá que se preocupar com o `Threshold Value` . Com o sensor calibrado, e o valor mínimo e máximo definidos, você poderá utilizar o valor default do `Threshold Value` , que é 50.

> **Observação**
>
> Os blocos não estão conectados em sequência apenas por uma questão visual, desta forma diminui-se a largura da tela ocupada pelo programa e facilita a visualização.

# Capítulo 27
# NXT – Utilizando o sensor de luminosidade

O kit EV3 contempla em um único sensor duas funcionalidades: cor e luminosidade. Já o kit NXT possuía sensores distintos, sendo um de cor e um de luminosidade. O bloco `Colour Sensor` do software LEGO® MINDSTORMS® Education EV3 funciona apenas com o próprio sensor do kit.

Entretanto, isso não um impedimento. Há um modo alternativo para que você possa utilizar o sensor de luminosidade em seu Brick NXT. Para controlar o sensor de luminosidade, utiliza-se o bloco `NXT Sound Sensor` .

## 27.1 O bloco NXT Sound Sensor

Figura 27.1: Ícone do bloco `NXT Sound Sensor`

Como já mencionado, este sensor encontrava-se presente no kit anterior, o NXT. Porém, é possível encontrá-lo, separadamente, para a compra.

Trata-se do bloco responsável pelo sensor de som. O sensor de som mede o nível de som, sendo que a medição pode ser feita em

dB ou dBa. Pode ser utilizado nos seguintes modos: `measure` , `compare` e `calibrate` .

- `Measure` : utilizado para medir o nível do som. Pode ser utilizado em:

  `-dB` : mede o nível do som na escala dB. Possui apenas uma saída: a `Sound Level` , que fornece o valor do nível do som na escala.

  `-dBa` : mede o nível do som na escala dBa. Também possui a mesma saída mencionada acima.

- `Compare` : modo utilizado para comparar o nível do som medido. Pode ser utilizado em qualquer uma das duas escalas.

- `Calibrate` : usado para calibrar o sensor. Seu funcionamento é semelhante ao do sensor de cor, que foi abordado logo no início. Com ele, pode-se calibrar o valor do nível máximo, mínimo e ainda reiniciar o sensor para as configurações originais.

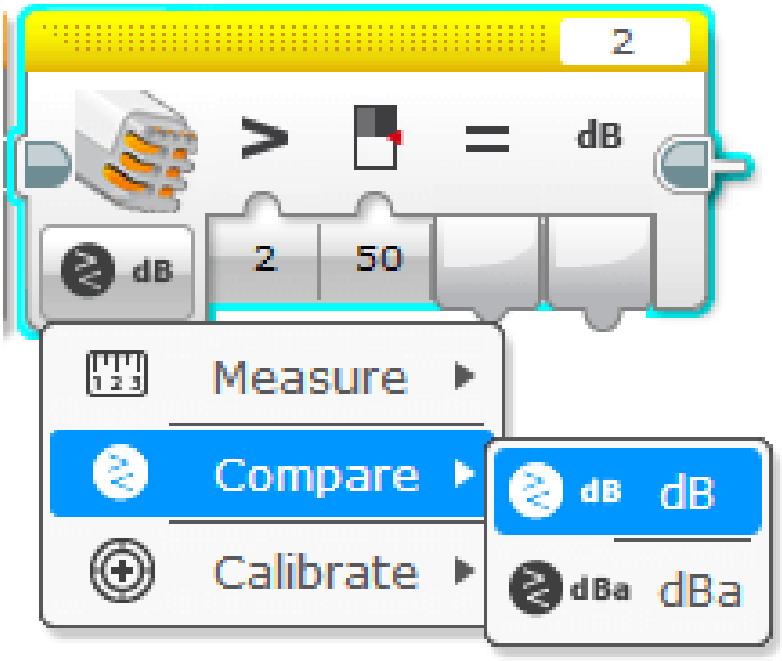



Figura 27.2: Bloco NXT Sound Sensor

**OBSERVAÇÃO**

dB (decibéis) e dBA são unidades para medida de som. Estas unidades medem a intensidade do som.

Parece estranho controlar o sensor de luminosidade utilizando um bloco responsável (originalmente) por controlar um sensor de som. Porém, o bloco `NXT Sound Sensor` apresenta um funcionamento semelhante ao bloco `Light Sensor` do software NXT.

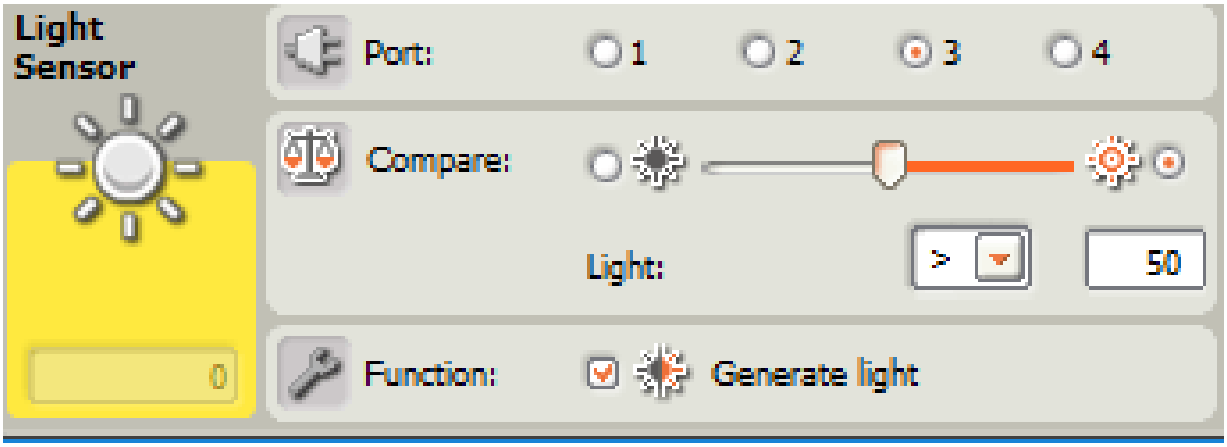



Figura 27.3: Bloco Light Sensor do software NXT – Configurações

Para que isso fique bem claro e você perceba mais nitidamente a semelhança, repare na constituição (entradas e saídas) do bloco `NXT Sound Sensor` e, no bloco `Colour Sensor` , ambos utilizados no modo de comparação.

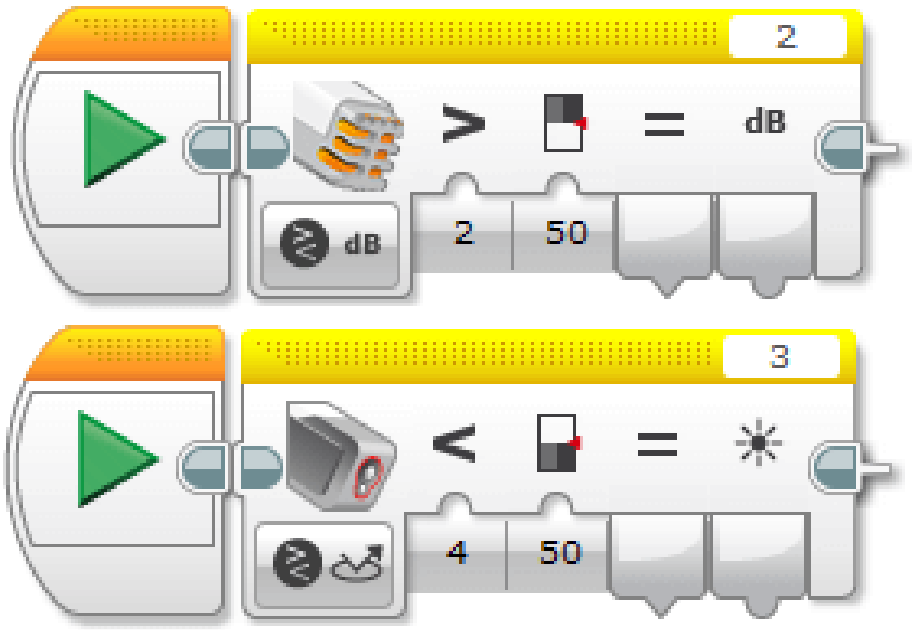



Figura 27.4: Bloco NXT Sound Sensor e Bloco Colour Sensor

A constituição dos blocos, ou seja, número de entradas e saídas, modos e todo o resto, é igual. A única coisa que muda são as

grandezas medidas.

## 27.2 Seguidor de linha NXT

Por exemplo, para criar um programa seguidor de linha, utiliza-se o bloco `NXT Sound Sensor` no modo `Compare - dB` . Veja como fica o programa seguidor de linha para um Brick NXT:

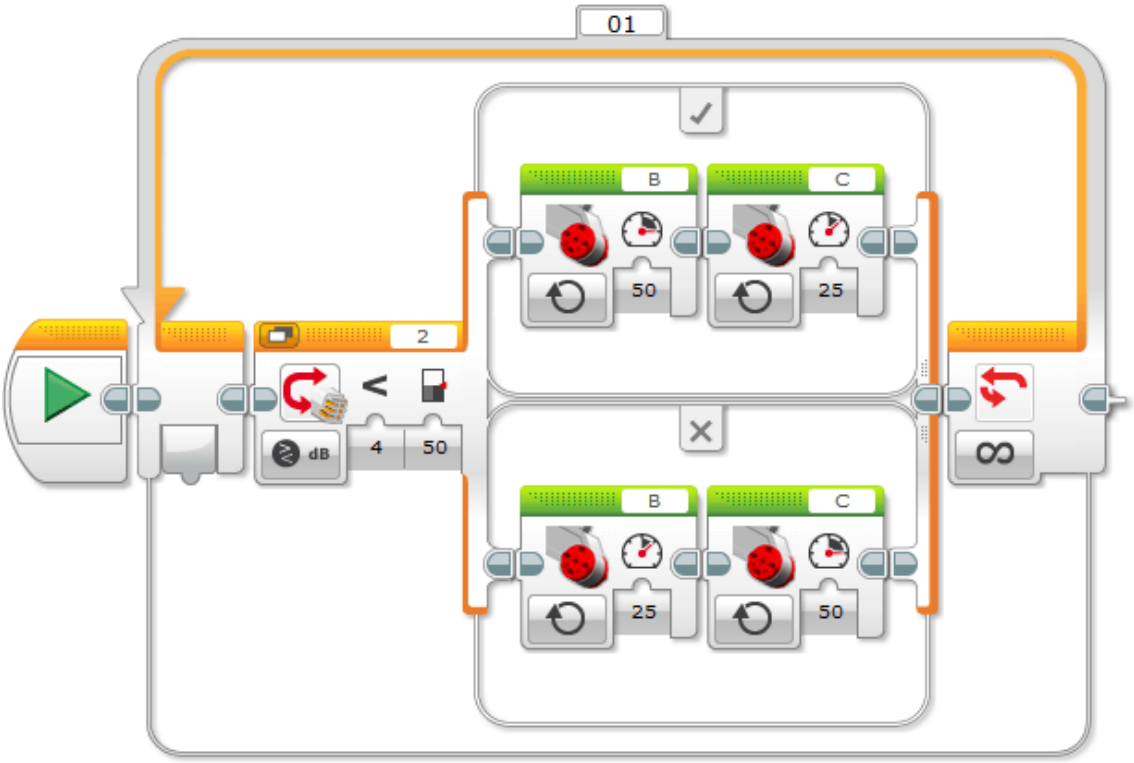



Figura 27.5: Seguidor de linha NXT

Estre programa pode ser acessado por meio do seguinte link:

https://github.com/vitor000x/ev3_Codes/blob/master/Recursos-Avancados/Programas/SL_NXT.ev3.

# Capítulo 28
# Aprimorando - Seguidor de Linha com bloco Wait

Neste capítulo, vamos voltar à primeira experiência apresentada no livro. Alguns conceitos novos serão demonstrados, e mais alguns blocos também. Trabalharemos uma forma mais interessante e inteligente para que o robô Seguidor de Linha faça um giro e termine seu movimento giratório exatamente na linha, sem sair da trajetória, assim ele poderá continuar seu trajeto de seguir a linha, realizar o giro, tudo isso indefinidamente.

A primeira mudança é que utilizaremos um bloco `Move Tank` , em vez de dois `Large Motor` , logo, configuraremos as duas potências diretamente em um bloco, já que o `Move Tank` combina dois blocos motores em um só. Esse `Switch` foi colocado dentro de um `Loop – Time Indicator` , configurado em 5 segundos de duração, que por sua vez foi colocado em um `Loop – Count` . Assim, após seguir a linha, ele realiza o giro (que programaremos a seguir) e após isso retorna ao movimento inicial: seguir a linha por 5 segundos, tudo isso por três vezes. Vale lembrar que é no parâmetro `Count` (apresentado quando o bloco está configurado no modo `Count` ) que definimos o número de vezes que o `Loop` será executado.

Observe como ficou o programa até aqui:

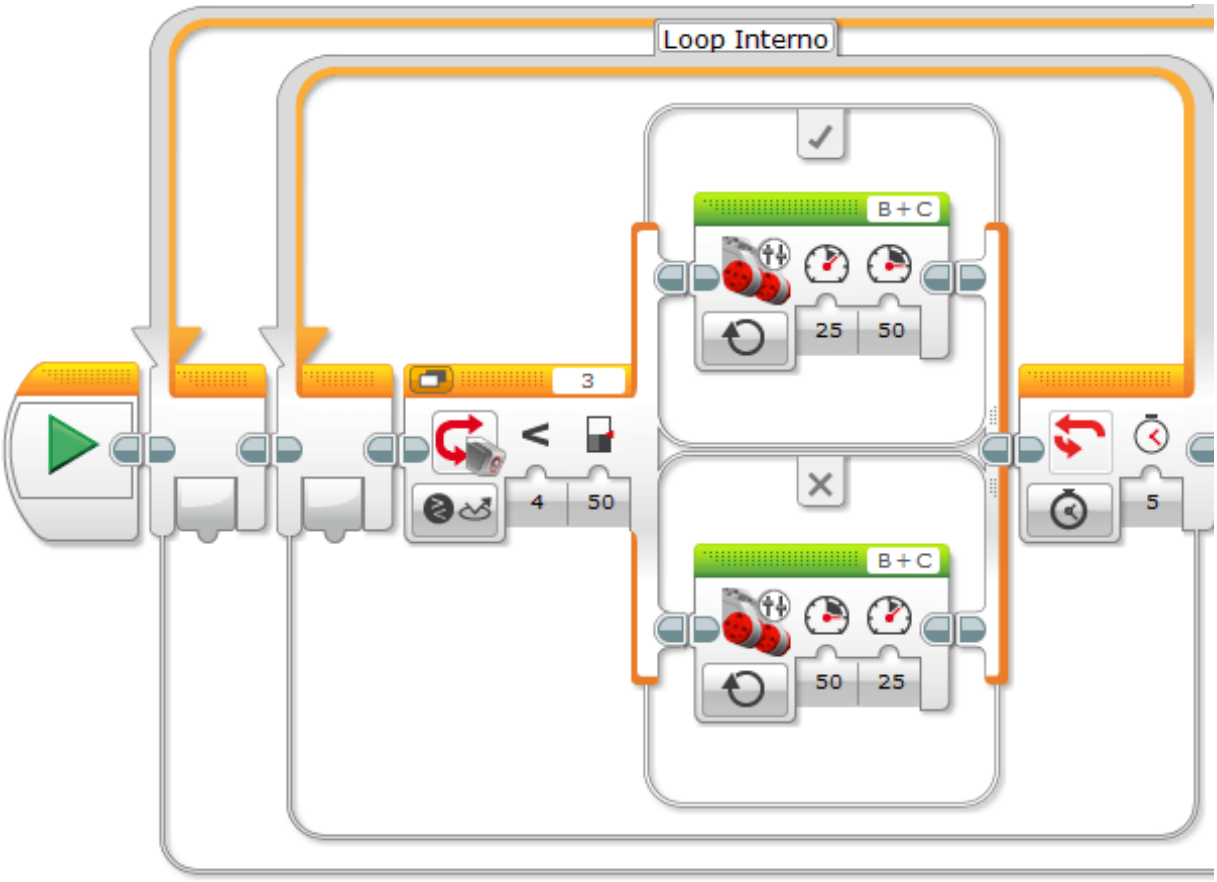



Figura 28.1: Primeira parte do programa: o robô segue a linha por 5 segundos

Perceba que o valor limiar, o `Threshold Value` do `Switch` , está como “50”, que é o valor `Default` . Isso porque foi feita uma calibração prévia do sensor, dessa forma, ele já tem os valores mínimo e máximo bem definidos. Esse é um dos benefícios de se realizar uma calibração no sensor.

Agora, faremos com que o robô faça o movimento de giro, dando uma volta, mas, sem que ele saia do traçado negro, ou seja, da linha que está seguindo. Para isso, vamos relembrar de um bloco muito importante, o bloco `Wait` . Ele serve para fazer com que nosso programa espere por algo, um determinado tempo, ou até mesmo uma condição, que será o nosso caso.

O primeiro passo será colocar o bloco responsável pelo movimento de giro. Para isso, utilizaremos o bloco `Move Tank` , no modo `On` ,

porém, ajustaremos um motor com potência positiva e outro com potência negativa. Desta forma, o robô girará em torno do próprio eixo. Em seguida, adicionaremos o bloco `Time Indicator` , configurado para 1 segundo. Assim, garantiremos que o movimento do bloco `Move Tank` (o movimento de giro) ocorra por pelo menos 1 segundo.

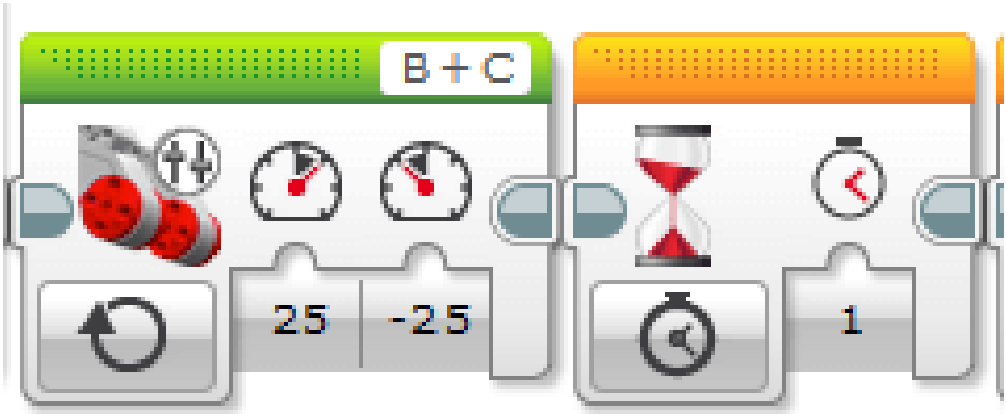



Figura 28.2: Bloco Move Tank, responsável pelo giro e, bloco Wait – Time Indicator

O penúltimo bloco que adicionaremos fará com que o robô só cesse o movimento de giro quando o sensor (e consequentemente o robô, pois o sensor está posicionado exatamente no meio do robô) estiver novamente sobre a linha negra. Para a realização dessa ação, utilizaremos o bloco `Wait` . Por meio desse bloco, diremos ao robô que ele só deve cessar movimento de giro quando estiver posicionado novamente sobre a linha negra, de forma que continue a segui-la sem sair do trajeto.

O bloco `Wait` deve ser configurado no modo `Colour Sensor` , desta forma, esperará que a condição estabelecida por nós para o `Colour Sensor` seja atendida, para que a próxima ação após este bloco seja executada.

E para que o robô pare exatamente na linha, o bloco `Wait` será utilizado com a seguinte configuração: `Colour Sensor – Compare – Reflected Light Intensity` . No parâmetro `Compare Type` , selecionamos a comparação `Less Than or Equal To` (menor que ou igual à) e no

parâmetro `Threshold Value` ajustamos com o valor 10 (pois, neste caso, a linha negra apresentou uma luminosidade inferior a 10).

Em seguida, é só adicionar o bloco `Move Tank` configurado no modo `Off` . Assim, quando ele para na linha negra, o motor `Move Tank` que estava realizando movimento de giro é desligado e o robô volta a seguir a linha.

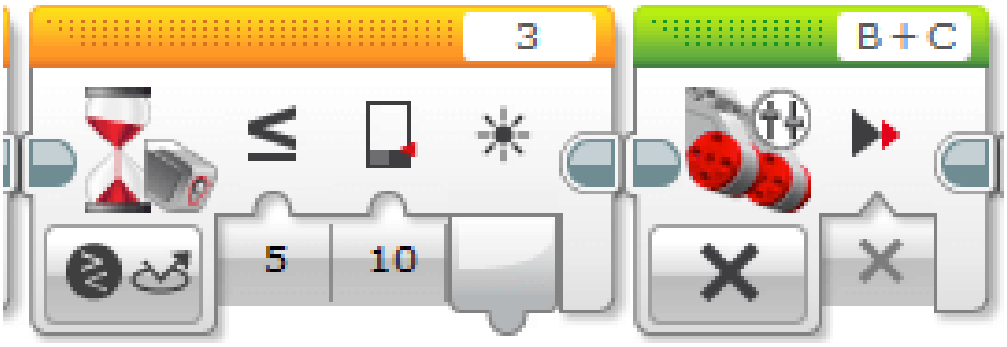



Figura 28.3: Bloco Wait – Colour Sensor – Compare - Reflected Light Intensity e Bloco Move Tank no modo Off

Observe o código completo:

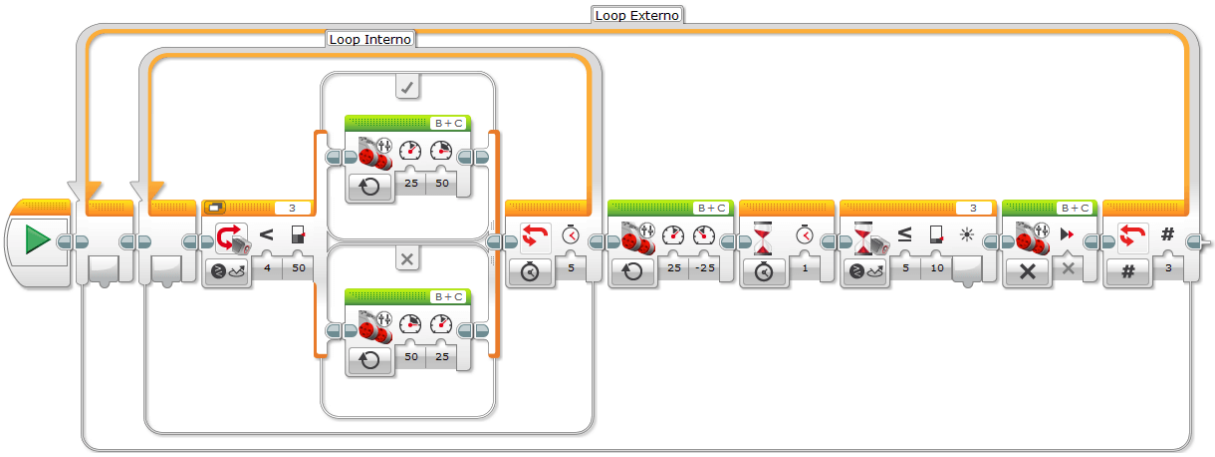



Figura 28.4: Bloco Wait – Colour Sensor – Compare - Reflected Light Intensity e Bloco Move Tank no modo Off

Com este simples incremento, o nosso robô agora executa um movimento muito mais preciso!

Por esta simples aplicação é possível ver o quão útil é o bloco `Wait` .

# Capítulo 29
# Aprimorando - Sumô com sensor ultrassônico

Na parte anterior a esta, criamos o robô Sumô, que tinha como objetivo empurrar um robô adversário, entretanto, havia um detalhe: isto devia ser feito sem que o robô ultrapassasse o perímetro delimitado. Contudo, o protótipo que utilizávamos não possuía nenhum modo de detectar um possível adversário.

Neste capítulo, faremos um aprimoramento do robô Sumô, adicionando a ele um sensor ultrassônico, para que, dessa forma, ele possa detectar seu adversário. O primeiro passo é realizar as alterações necessárias no protótipo, seguindo estas instruções:

https://github.com/vitor000x/ev3_Codes/blob/master/Recursos-Avancados/Instructions/SUS.pdf.

O primeiro bloco inserido deve ser o bloco `Loop` , para que o programa seja executado infinitamente. O segundo bloco é `Switch` , configurado no modo `Colour Sensor – Compare – Reflected Light Intensity` , pois assim controlaremos nosso programa com base no valor medido pelo sensor de cor (o mesmo funcionamento que vimos antes). O `Switch` está configurado no modo `Less Than` e com valor 50 para o `Threshold Value` .

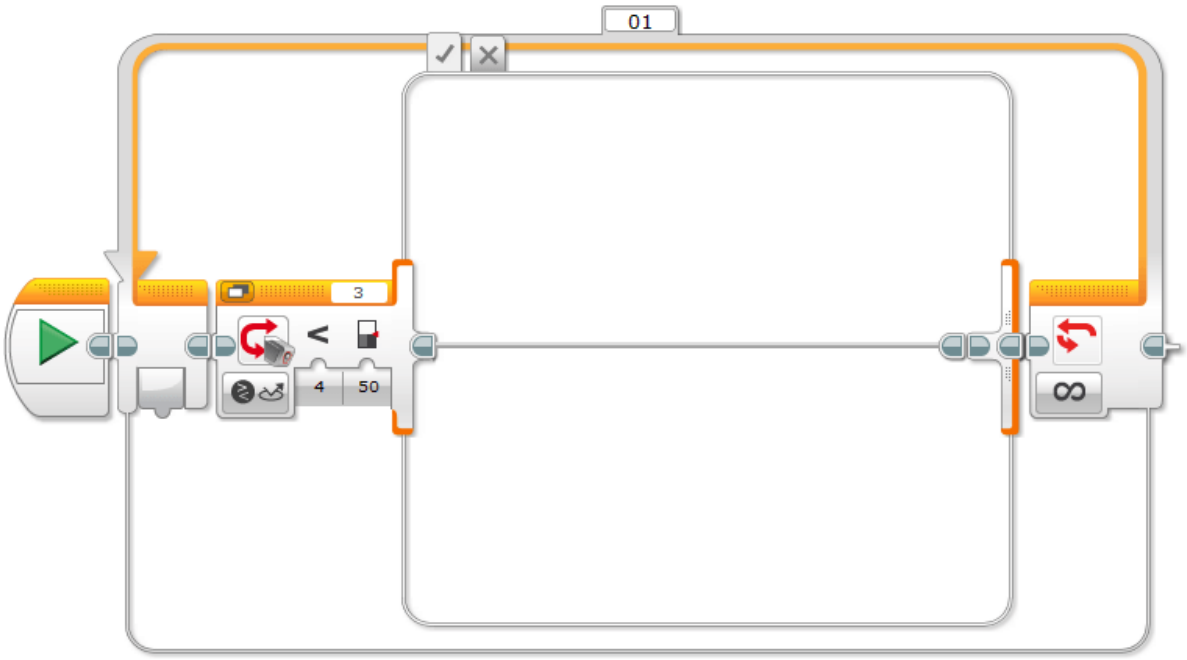



Figura 29.1: Sumô: detectando invasores - Estrutura do programa

No `Switch` , temos o `Case True` e o `Case False` . Inseriremos instruções para os dois casos. O programa executará o `Case True` quando a luminosidade for menor que 50, ou seja, quando detectar a linha negra. Para este caso, inseriremos dois blocos `Move Tank` . O primeiro bloco `Move Tank` , será no modo `Off` . O segundo bloco, inseriremos no modo `On for Rotations` , e ajustaremos seus parâmetros de potência com valor `-50` . Este bloco deve ser executado apenas por uma rotação.

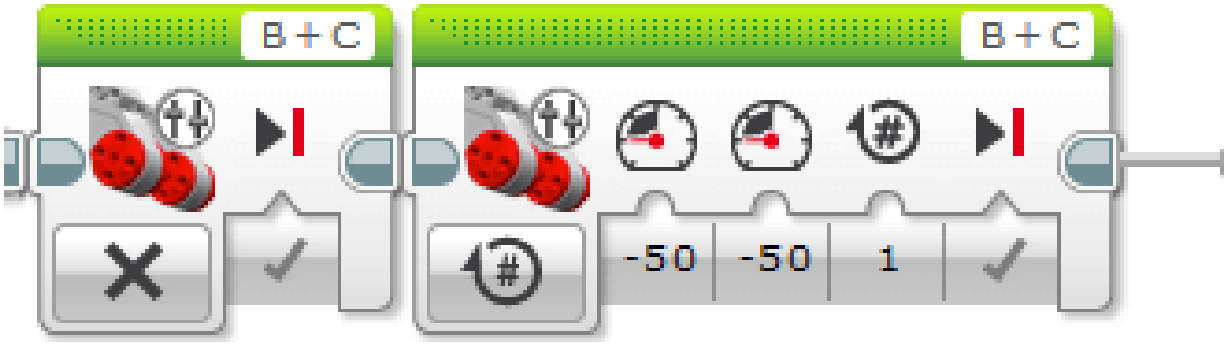



Figura 29.2: Sumô: detectando invasores - Switch - Case True

A sequência de blocos que inserimos tem a seguinte função: quando o robô detectar a linha negra (lembrando que a linha negra delimita a arena e o robô não pode ultrapassar este limite) ele deve parar os motores, cessar seus movimentos e, então, realizar um movimento de giro; dessa forma ele voltará à parte interna da arena, a parte branca.

O passo seguinte é programar a sequência de instruções para o case `False` do `Switch` , executado enquanto o robô estiver dentro da arena (parte branca). Para este `Case` inserimos outro `Switch` , que, por sua vez, terá a função de controlar o sensor ultrassônico para detectar a presença do robô adversário.

O `Switch` deve ser inserido no modo `Ultrasonic Sensor – Compare – Distance Centimetres` . Escolhemos o tipo de comparação `Less Than` e ajustamos o `Threshold Value` para 20 centímetros.

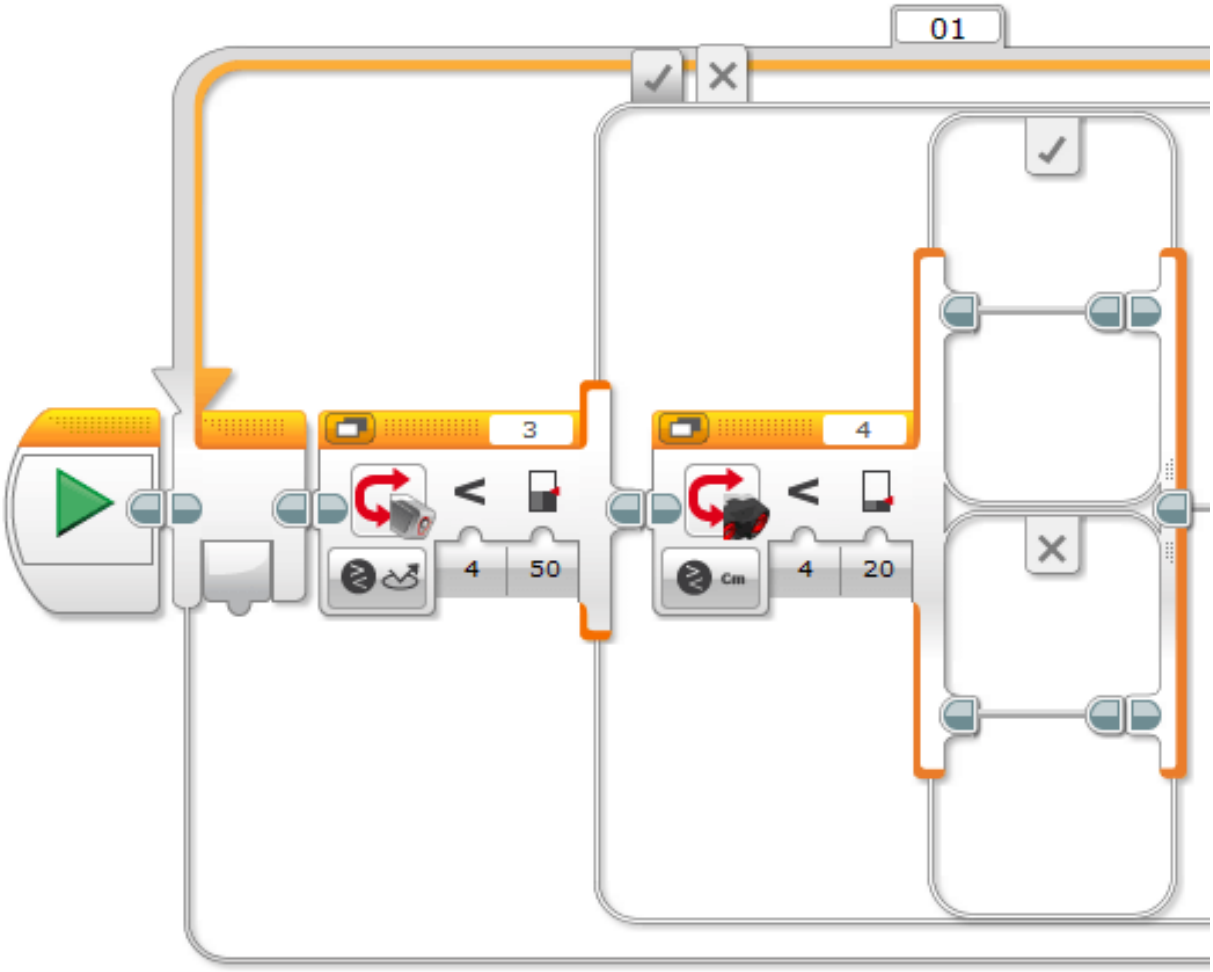



Figura 29.3: Sumô: detectando invasores - Switch - Ultrasonic Sensor

Quando a distância comparada pelo sensor ultrassônico for menor que 20 centímetros, isso significa que o robô adversário está a uma distância menor que 20 cm, portanto, programaremos o robô para avançar na direção de seu oponente, dessa forma, ele o empurrará. Para isso, é só inserir o bloco `Move Tank` , configurado no modo `On` . Para que o robô obtenha um maior êxito, utilizamos potência total nos dois motores.

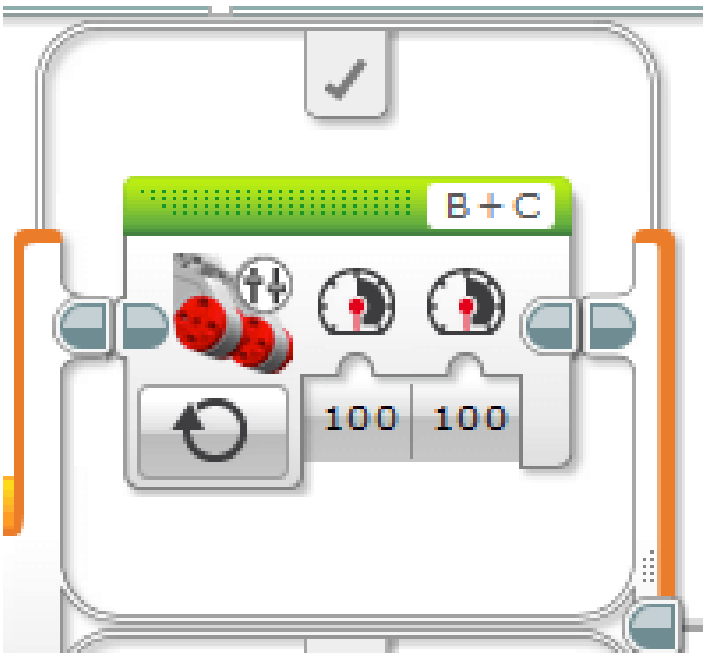



Figura 29.4: Sumô: detectando invasores - Switch - Ultrasonic Sensor - Case True

O `Case False` do `Switch` será executado quando a distância for maior que 20 cm (e por estar encapsulado em outro `Switch` , quando a luz refletida for maior que 50). Para este caso, inseriremos outro bloco `Move Tank` , também configurado no modo `On` , porém, para ele, utilizaremos valores médio de potências (50 para cada um dos motores). Mas, aqui, vamos inserir um valor de potência negativo para o motor B, `-50` , enquanto que, o motor C conterá o valor `50` . Com essa configuração, o robô executará um movimento giratório infinito até sair desse `Case` . Assim, o robô fica girando, somente, à espera do robô adversário.

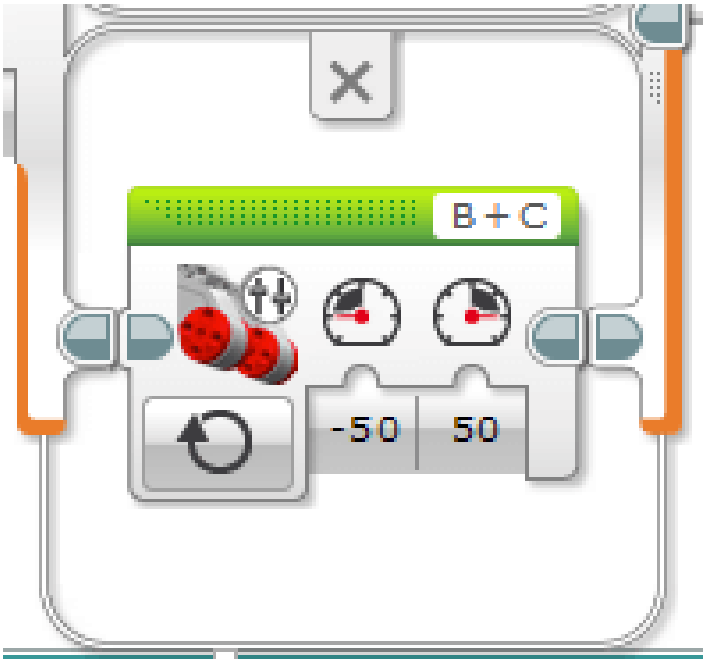



Figura 29.5: Sumô: detectando invasores - Switch - Ultrasonic Sensor - Case False

Observe como ficou o programa:

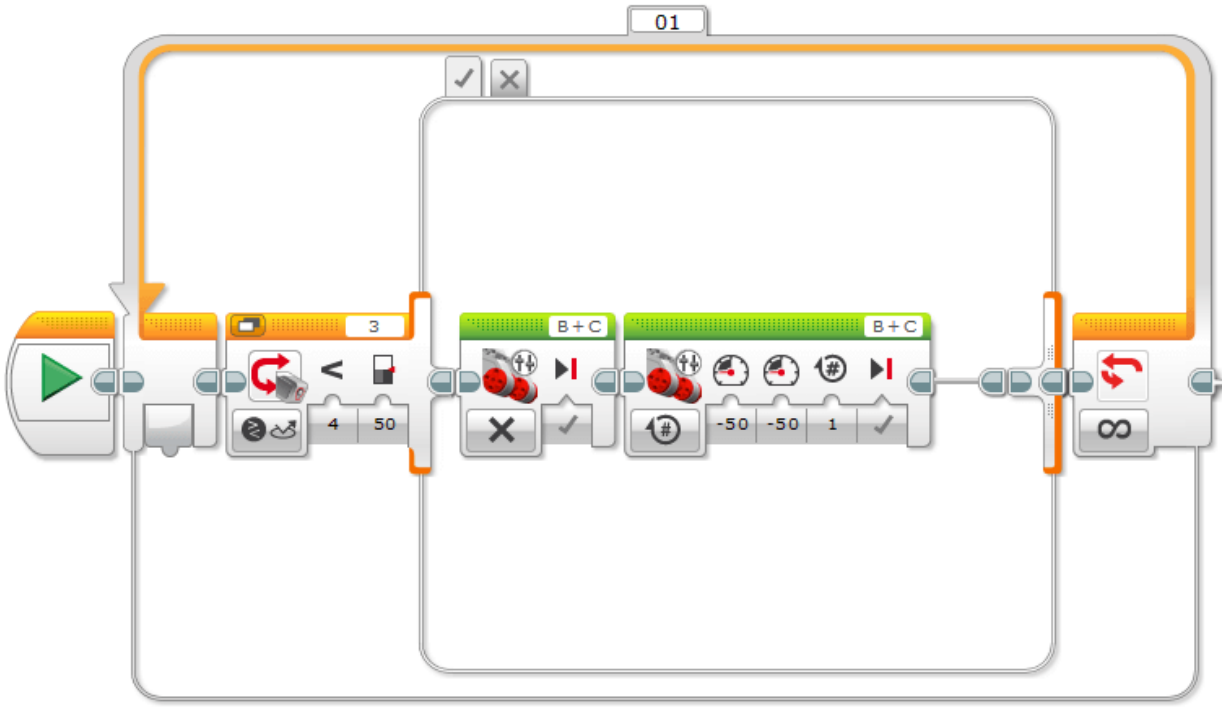



Figura 29.6: Sumô: detectando invasores - Código completo - Case True

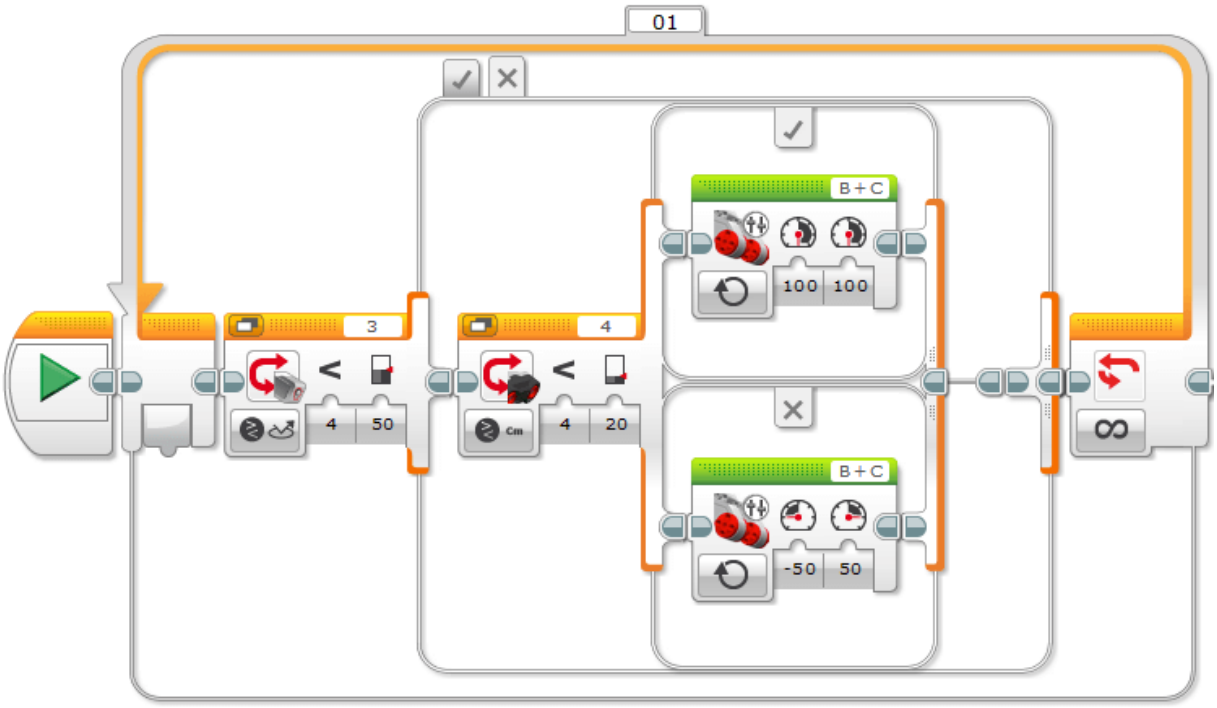



Figura 29.7: Sumô: detectando invasores - Código completo - Case False

Você pode acessar o programa no seguinte link:

https://github.com/vitor000x/ev3_Codes/blob/master/Recursos-Avancados/Programas/SM-U.ev3

## 29.1 Utilizando dois sensores

Para aumentar a eficiência e tenha um melhor funcionamento dentro do ringue, é recomendável construí-lo utilizando dois sensores de cor. Para isso, basta adicionar mais um sensor na outra extremidade do robô.

No mesmo link anterior, você pode conferir como ele ficou. O código para um robô Sumô que utiliza dois sensores é bem simples, apenas algumas alterações são necessárias.

Nosso primeiro passo é alterar o tipo do `Switch` ; nenhuma modificação do código contido no `Switch` será necessária. Alteramos o `Switch` para o modo `Logic` ; mais adiante você entenderá o motivo.

Como temos dois sensores de cor, é necessário obter a intensidade medida pelos dois sensores. Por isso, adicionamos dois blocos `Colour Sensor` no modo `Compare – Reflected Light Intensity` . Ajustamos esses blocos da mesma forma que antes. Observe na imagem a seguir, como está o programa.

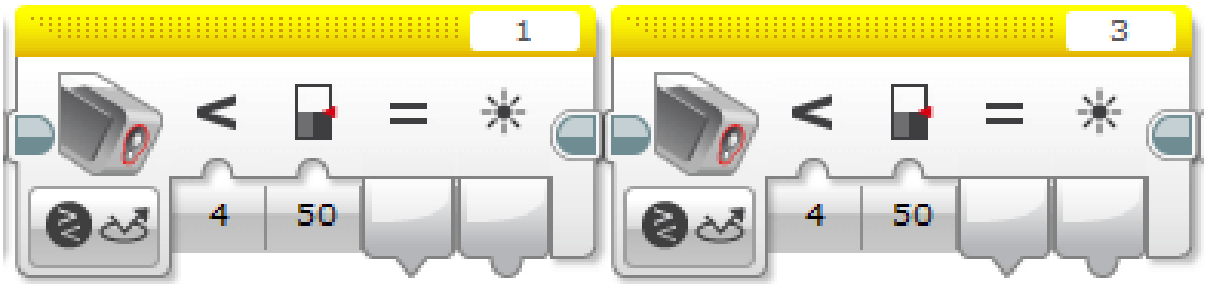



Figura 29.8: Bloco Colour Sensor

Agora que já temos os valores dos dois sensores, é necessário adicionar um bloco que será responsável por comparar a condição dos blocos, dessa forma, caso um sensor registre a intensidade de luz refletida menor que 50, o `Switch` será acionado. Para fazer isto, utilizamos o bloco `Logic Operations` .

Adicionamos o bloco `Logic Operations` no modo `Or` , pois nesse modo, basta que um sensor esteja com valor lógico `True` (ou seja, ele está medindo uma luminosidade refletida menor que 50) para que o bloco `Logic Operations` gere um valor lógico `True` .

Para finalizar, é só realizar a conexão entre os blocos. Enviamos o resultado das operações realizadas pelos blocos sensores às entradas do bloco `Logic Operations` e, por fim, conectamos a saída do bloco `Logic Operations` à entrada lógica do `Switch` .

Desse modo, sempre que um sensor detectar a linha negra que delimita a arena, o `Switch` será ativado e, então, o robô executará

todas as ações devidas.

A seguir você pode conferir as modificações realizadas no código.

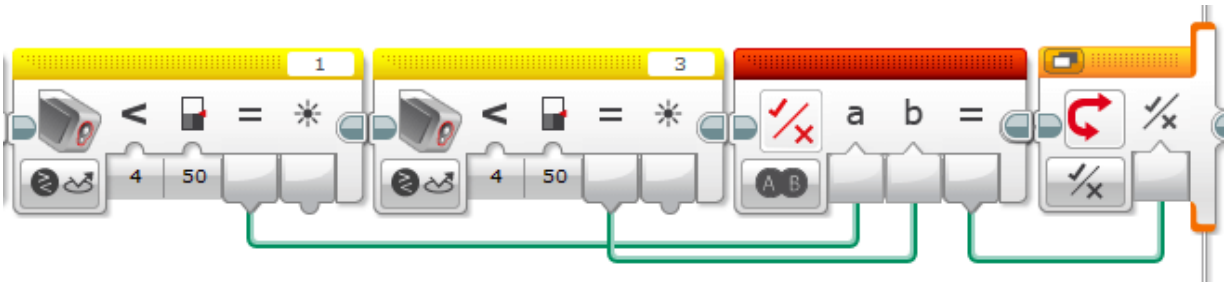



Figura 29.9: Sumô com dois sensores de cor

No seguinte link você pode encontrar o programa feito aqui:

https://github.com/vitor000x/ev3_Codes/blob/master/Recursos-Avancados/Programas/SM-D.ev3

# Capítulo 30
# Aprimorando - Explorer com o Gyro Sensor

No capítulo referente ao robô Explorer, uma questão ficou pendente: como realizar um giro perfeito para a direita ou para a esquerda, um giro de exatos 90 graus. Anteriormente, não se teve uma preocupação com o giro do robô, e a única forma que havia para melhorar o giro era testando diferentes valores de rotação, de tempo, diferentes modos etc. Antes, só conhecíamos a forma empírica. Neste capítulo será apresentada uma solução bastante simples e eficaz para essa questão.

No Seguidor de Linha, visto nesta parte, adotamos uma estratégia para que o robô realizasse uma volta sem sair da trajetória utilizando o bloco `Wait` . Aqui também utilizaremos o bloco `Wait` , porém, vamos aprender a utilizar um sensor muito importante no kit, o Gyro Sensor. Com o Gyro Sensor é possível programar o robô para girar exatamente na quantidade de graus desejada.

No seguinte link você encontra as instruções necessárias para acoplar o Gyro Sensor ao seu robô:

https://github.com/vitor000x/ev3_Codes/blob/master/Recursos-Avancados/Instructions/EGS.pdf

Agora, vamos conhecer um pouco mais sobre o bloco `Gyro Sensor` .

## 30.1 O bloco Gyro Sensor

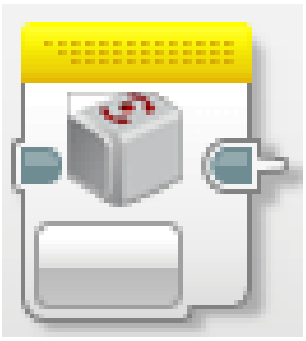


Figura 30.1: ícone do bloco Gyro Sensor

Bloco responsável pelo sensor de giro. Possui três modos de funcionamento: `measure` , `compare` e `reset` . As medições sempre são realizadas em relação ao eixo do sensor.

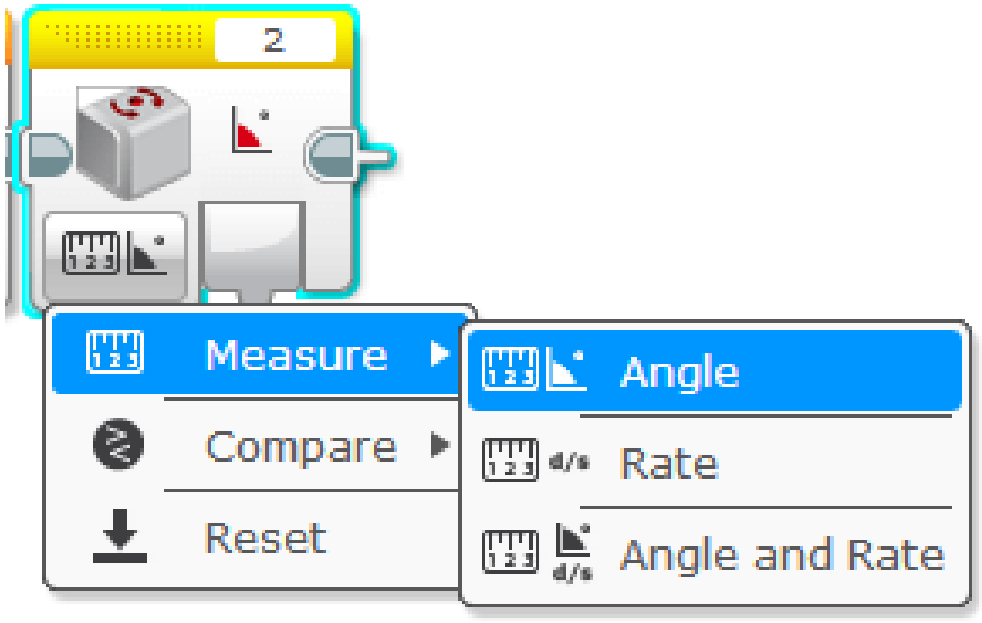



Figura 30.2: Bloco Gyro Sensor

- `Measure` : utilizado para medir a taxa de rotação em relação a seu próprio eixo (representado pela bolinha vermelha central no sensor), os valores negativos indicam o lado esquerdo, e os positivos o lado direito. A taxa de rotação pode ser medida das seguintes formas: `Angle` , `Rate` e `Angle and Rate` .

O modo `Angle` mede o ângulo de rotação em graus. A saída de mesmo nome, `Angle` , contém o valor medido.

O modo `Rate` mede a taxa de rotação em graus por segundo. Na saída `Rate` , obtém-se o resultado da medição.

O modo `Angle and Rate` mede o ângulo de rotação e a taxa de rotação ao mesmo tempo. Logo, possui duas saídas: `Angle` e `Rate` , nas quais se encontra o valor medido.

- `Compare` : utilizado para comparar a taxa de rotação ( `Rate` ) ou o ângulo de rotação ( `Angle` ).
- `Reset` : reinicia o sensor, voltando a apresentar o valor de ângulo do sensor em 0°.

Agora que você já viu um pouco mais sobre o bloco `Gyro Sensor` , vamos ao código. No programa não utilizaremos o bloco `Gyro Sensor` propriamente, porém, utilizaremos o bloco `Wait` configurado no modo `Gyro Sensor` , por isso, entender o funcionamento deste bloco é imprescindível.

**FIQUE ATENTO!**

Antes de realizar a experiência, lembre-se de que é muito importante estar atento à conexão das portas. Por exemplo, se o seu motor C corresponder ao motor direito e, o motor B, ao lado esquerdo, para realizar o giro à direita, configure o parâmetro `Steering` com valor 100. E, para o giro à esquerda, utilize o valor -100. Se você não se lembra muito bem de como funciona o movimento dos motores, releia a seção “Movimentos” do capítulo *19. Seguidor de Linha*.

Utilizaremos o código do Explorer que criamos anteriormente, desse modo não precisamos reescrever todo o código, até porque, a única coisa que modificaremos é o `Switch` que contém os blocos responsáveis pelo movimento do robô.

Vamos criar agora o movimento responsável pelo giro à direita. O giro para a esquerda consiste na mesma sequência de blocos, porém, com alguns valores de parâmetros negativos (vimos que utilizamos números negativos para realizar movimentos à esquerda).

O primeiro bloco que inseriremos será o bloco `Move Steering` , configurado no modo `On` , com potência ajustada para 25 por cento (este valor é opcional), e no parâmetro `Steering` ajustaremos para a rotação máxima, ou seja, 100. Até agora, se executássemos esse programa o robô iria girar para sempre, pois não definimos quando ele deve parar, apenas definimos que ele deve girar.

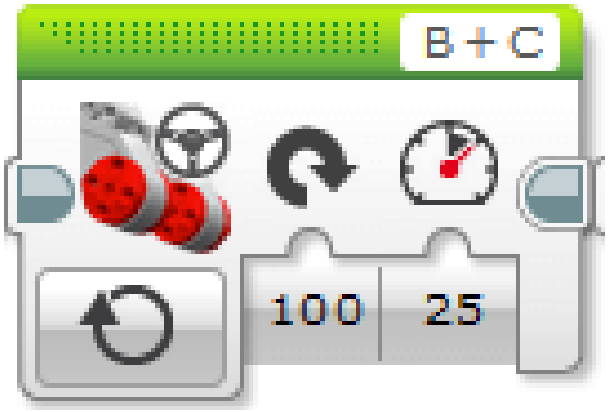



Figura 30.3: Bloco Move Steering - Movimento para a direita

Conforme visto anteriormente, o bloco `Wait` faz com que o robô espere até que uma ação aconteça, para realizar uma próxima sequência de ações. Dessa forma, utilizando o bloco `Wait` no modo `Gyro Sensor` diremos ao robô que ele precisa esperar até que o `Gyro Sensor` meça 90 graus; quando isso acontecer, quando o giro tiver sido de 90 graus, então, ele deve desligar os motores.

Para isso, logo após o bloco `Move Steering` , inserimos o bloco `Wait` configurado no modo `Gyro Sensor – Change – Angle` . Com essa configuração, estamos dizendo ao programa que ele deve esperar até que o ângulo mude e, aí, execute a próxima ação; que no nosso caso será desligar os motores.

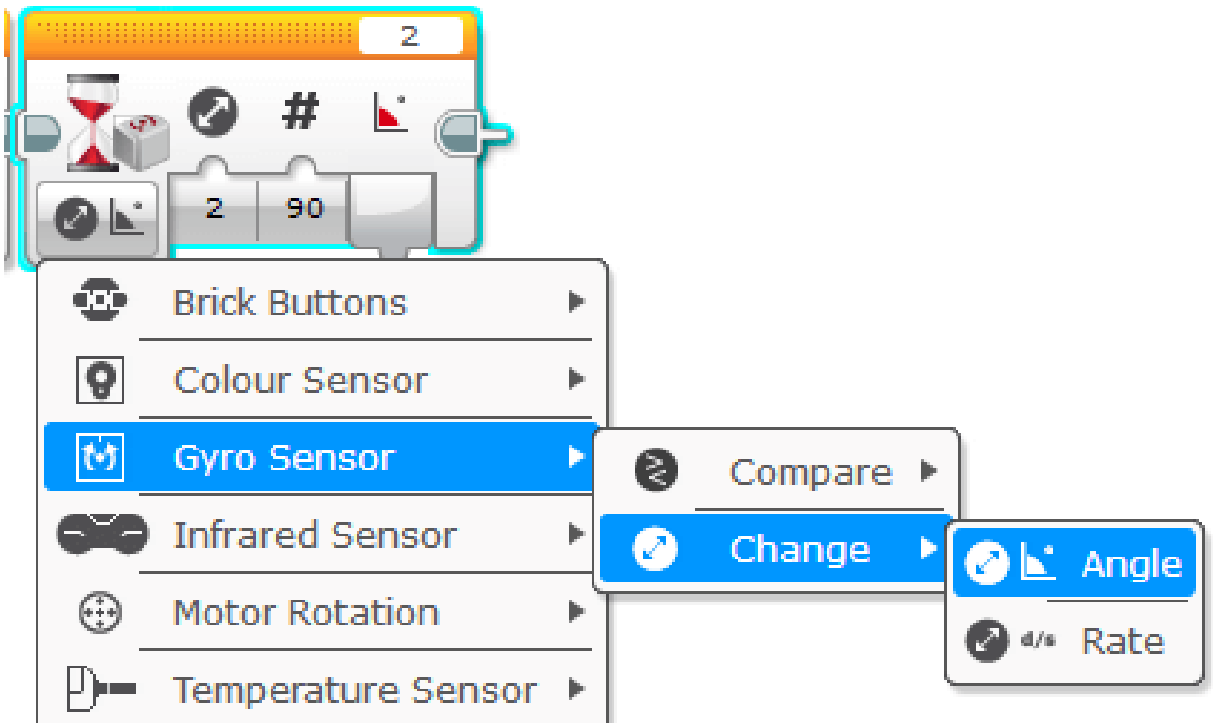



Figura 30.4: Bloco Wait - Gyro Sensor - Change - Angle

Agora, o próximo passo consiste em configurar os parâmetros do bloco para que ele espere até que a rotação seja de 90 graus. No bloco há duas entradas e uma saída. Configuraremos apenas as entradas, para que o bloco opere de acordo com nossas necessidades. A saída, que é `Angle` , fornece o ângulo da rotação.

No parâmetro `Direction` é definido o que esperaremos na direção, se ela deve aumentar ( `Increase` ), diminuir ( `Decrease` ) ou pode ser qualquer um dos dois anteriores ( `Any` ). O parâmetro `Amount` recebe o valor que queremos para a direção, neste caso, o valor do ângulo, 90 graus. Com isso, estamos dizendo ao robô que ele deve esperar por uma modificação na direção do robô, que seja igual a 90 graus.

Ao configurar o bloco `Move Steering` com o valor positivo (a escala positiva corresponde aos movimentos para a direita), o robô executará o movimento de giro à direita e, quando a mudança na direção for equivalente a 90 graus, o bloco `Wait` terá sua condição de espera satisfeita, dando fim a ela. Logo, a próxima ação pode ser

executada. E, como o robô já executou o giro para a direita, podemos desligar os motores, finalizando o movimento. Para isso, inserimos o bloco `Move Steering` no modo `Off` .

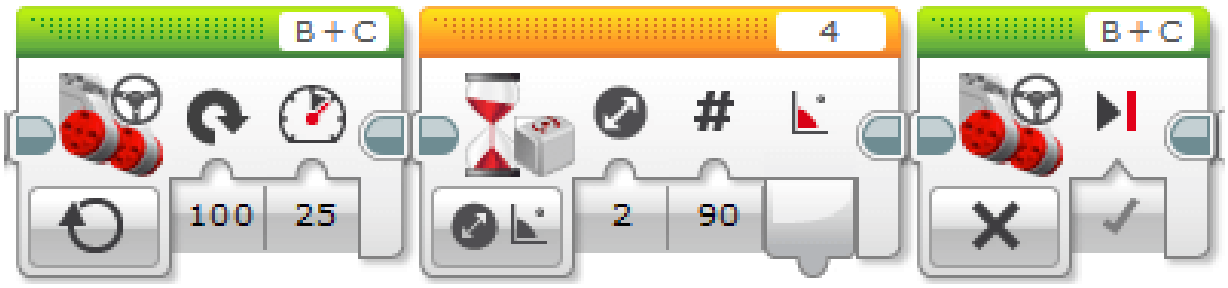



Figura 30.5: Movimento para a direita

Para realizar o movimento para esquerda, repita a mesma sequência de blocos. Porém, como o movimento é para a esquerda, altere o valor do parâmetro `Steering` para -100. Dessa forma, o robô fará o movimento para a esquerda.

Figura 30.6: Movimento para a esquerda

Aqui está a sequência de blocos completa:

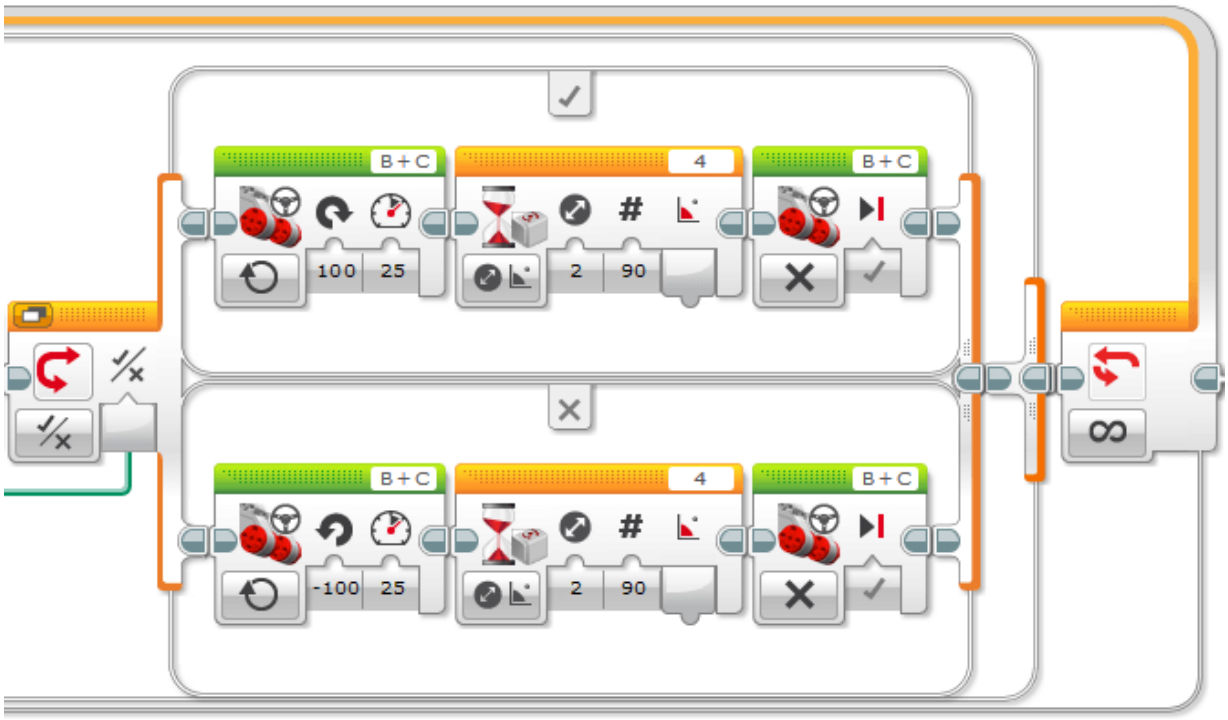



Figura 30.7: Switch responsável pelo movimento de giro

Para ter acesso ao programa, clique no link:

https://github.com/vitor000x/ev3_Codes/blob/master/Recursos-Avancados/Programas/E-GS.ev3

# Capítulo 31
# Conclusão

Mais uma etapa finalizada. O conteúdo didático do livro se encerra por aqui. Esta parte do livro teve como principal objetivo apresentar dicas e recursos para que você possa evoluir ainda mais na sua jornada pela programação.

Vimos conceitos e práticas muito úteis e indispensáveis para sua aprendizagem e que constituem um diferencial para o seu currículo. Desde os primeiros capítulos, até aqui, você já adquiriu uma boa bagagem em programação. Durante esse percurso, muitas ferramentas, recursos e blocos foram vistos. Com certeza, você deve ter se tornado um programador bem melhor, que já sabe desenvolver suas próprias modificações e melhorias nos programas apresentados.

Porém, o livro não acaba por aqui, ainda. Na próxima parte (a última) estão minhas últimas dicas e sites para que você possa explorar ainda mais as possibilidades do seu kit LEGO MINDSTORMS e absorver ainda mais conhecimento.

# E AGORA?

O livro acaba por aqui, mas suas possibilidades não. Os kits robóticos LEGO® MINDSTORMS® contam com inúmeras funcionalidades e recursos. Este livro teve como objetivo fazer uma introdução ao mundo da robótica com os kits robóticos da linha LEGO® MINDSTORMS®, e assim foi feito. Pensando nos seus próximos passos, estas últimas páginas visam fornecer um pequeno guia para que você possa continuar a explorar as possibilidades do seu kit.

## Capítulo 32
## Sites para seguir seus estudos

Apesar de já possuir bastantes adeptos no Brasil, ainda há pouco conteúdo a respeito dos kits robóticos da linha LEGO® MINDSTORMS®. Mas há um conteúdo muito grande de projetos, artigos e tutoriais em inglês. Por isso, separamos alguns sites, bem bacanas, para que você possa encontrar ainda mais informações, inspirações e ideias para seus projetos futuros.

- **Site oficial da LEGO®** (https://www.lego.com/en-us/mindstorms): para os usuários da Home Edition, (a página ainda não foi traduzida para o português) este é um ótimo ambiente de aprendizagem. Nele você encontrará vídeos, tutoriais e mais informações.
- **LEGO® Engineering** (http://www.legoengineering.com/): um site americano muito interessante e riquíssimo em conteúdo sobre os kits robóticos da linha LEGO® MINDSTORMS®. Lá você encontrará desde o conteúdo mais básico até tópicos mais avançados.
- **Robot Square** (http://robotsquare.com/): aqui você encontra tutoriais, projetos e artigos. O site também possui outros conteúdos sobre robótica, além dos conteúdos sobre a linha

LEGO® MINDSTORMS®, como notícias, guias de produtos etc. Vale a pena conferir.

- **Danny's LAB** (https://robotics.benedettelli.com/): é um site incrível também. Nele você encontra muitos modelos de robôs para construir. Contudo, o conteúdo não é gratuito: lá você encontrará as informações para adquirir os livros, que contêm projetos incríveis. Vale ressaltar também que o conteúdo está em inglês.
- **Builderdude35** (http://builderdude35.com/): é o site do Kyle Markland, aliás, ele é conhecido pelo apelido de Builderdude35, que justamente é o nome de seu site. No site dele você encontra diversos tutoriais de robótica.

**Lessons Plans**

A LEGO® disponibiliza Lessons Plans para os professores e educadores. Esse conteúdo apresenta, por meio de um roteiro, tudo o que é necessário para desenvolver as atividades propostas, contendo objetivos, habilidades requeridas e mais informações. Você também pode filtrar a busca de projetos por nível de complexidade.

Para saber mais sobre os Lessons Plans acesse:

https://education.lego.com/en-us/lessons?pagesize=12&Products=LEGO%20MINDSTORMS%20Education%20EV3%20Core%20Set/.

Nesta página você encontrará todos os planos de aula disponibilizados pela LEGO®. Apesar de ser um recurso voltado especialmente para professores e educadores, nada impede que os demais leitores possam tirar proveito de seu conteúdo.

# Capítulo 33
# Competições

No Brasil existem algumas competições de robóticas, a maioria delas voltada para estudantes do ensino fundamental, e algumas abrangem também o ensino médio. Pensando em um possível interesse do leitor em alguma competição, listamos aqui algumas delas.

- **OBR** - Segundo a informação do próprio site, a "Olimpíada Brasileira de Robótica (OBR) é uma das olimpíadas científicas brasileiras que se utiliza da temática da robótica". Para saber mais acesse: http://www.obr.org.br/.
- **TJR** – O "Torneio Juvenil de Robótica é um evento gratuito de abrangência nacional que ocorre desde 2009, para difusão da Robótica no ambiente escolar da Educação Básica até a Educação Superior". O site oficial é o seguinte: http://www.torneiojrobotica.org/.
- **FIRST® LEGO®** - "O Torneio de Robótica FIRST® LEGO® League é um programa internacional de exploração científica, projetado para fazer com que crianças e jovens de 9 a 16 anos se entusiasmem com ciência e tecnologia e adquiram habilidades valiosas de trabalho e de vida". Mais informações você encontra no site: http://www.portaldaindustria.com.br/sesi/canais/torneio-de-robotica/.

Essas competições são uma ótima forma de testar seus conhecimentos e desenvolver suas habilidades. Algumas modalidades envolvem maiores habilidades, mas não é nada que bons treinos e algumas horas de estudo e dedicação não resolvam!

# Capítulo 34
# Explorando

## 34.1 LEGO® MINDSTORMS® Education EV3

O próprio software LEGO MINDSTORMS® Education EV3 é uma ótima ferramenta de aprendizagem. Na IDE está disponibilizado um enorme conteúdo para que você possa explorar todas as possibilidades do seu kit. Se você deseja construir mais robôs e quer contar com manuais de instruções para isso, basta acessar o `Building Instructions` ; a opção `Core Set Models` fornece um bom número de robôs para você tentar.

Na opção `Expansion Set Models` estão os robôs que possuem um grau maior de complexidade e são mais desafiadores. Porém, eles necessitam de um kit de expansão. O LEGO® MINDSTORMS® Education EV3 Expansion Set é vendido separadamente e, se você tem interesse, é uma boa aquisição. Se você se interessou, acesse o link para saber mais: https://education.lego.com/en-us/products/lego-mindstorms-education-ev3-expansion-set/45560.

### Data Logging

Um recurso muito importante do software LEGO® MINDSTORMS® Education EV3 é a ferramenta `Data Logging` (que foi brevemente mencionada no livro). Ela consiste em um importante recurso adicionado à linha EV3. Com ele, você pode gravar, ler e analisar os dados dos sensores e, dessa forma, deixar seus programas cada vez mais precisos. Se você deseja aprimorar suas habilidades e melhorar cada vez mais a precisão e confiabilidade de seus programas, essa é uma ferramenta importantíssima.

## 34.2 LEGO® MINDSTORMS® EV3 Home Edition

Conforme abordamos, uma das principais vantagens da versão Education são suas possibilidades. O software da versão Education apresenta muito mais recursos e ferramentas. Então, se você é usuário da versão Home Edition e ainda utiliza o software LEGO® MINDSTORMS® EV3 Home Edition, talvez seja hora de mudar para o LEGO® MINDSTORMS® Education EV3. Como dito no início do livro, a única vantagem do kit Home Edition são os modelos de robôs que contam com instruções de montagem, além de o kit possuir um "controle remoto", para controlar o sensor infravermelho. Logo, seu foco principal são as crianças que estão começando a programar, até mesmo pelo seu visual mais chamativo e robôs que se parecerem com figuras como: animais, humanoides etc.

# Capítulo 35
# Novas linguagens e ambientes de programação

Apesar de os softwares LEGO® MINDSTORMS® (ambas as versões) serem as ferramentas oficias para a programação do Brick, existem outras ferramentas que podem ser utilizas para a programação do kit. Conhecer novas linguagens e novas plataformas é altamente benéfico, pois amplia seus horizontes e, talvez, a nova plataforma ofereça uma curva de aprendizagem melhor, ou até mesmo mais ferramentas e recursos para a programação. De qualquer forma, é sempre bom aprender uma nova linguagem. Por esses motivos, estão listadas aqui algumas linguagens/plataformas que você pode aprender e utilizar na criação de programas para o seu kit.

## 35.1 Microsoft Make Code: LEGO® MINDSTORMS® EV3

https://makecode.mindstorms.com/

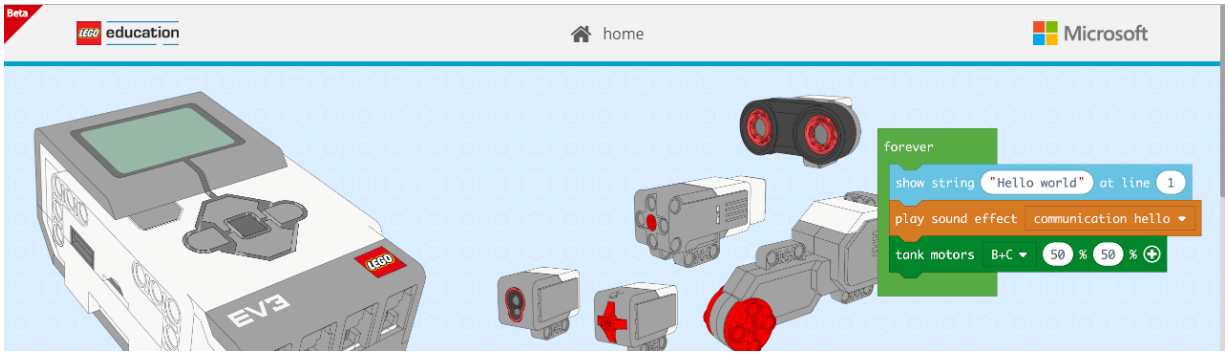



Figura 35.1: Microsoft Make Code: LEGO® MINDSTORMS® EV3

O Microsoft Make Code é um lugar que reúne ambientes de programação para várias tecnologias que possuem o objetivo de

ensinar programação de forma inovadora e divertida, dentre elas o kit LEGO® MINDSTORMS® EV3. É uma plataforma livre ao acesso de todos, totalmente gratuita, mas em inglês.

Acessando o site, logo na página inicial, você encontra todas as informações de que precisa para começar a programar seu EV3 utilizando a plataforma. Todas essas informações você encontra nas seguintes seções, que estão ao longo da página: *Getting Started*, *Tutorials*, *Design Engineering*, *Coding* e *Maker*. Essas seções contêm informações relacionadas ao hardware e ao software. Para criar um novo projeto basta clicar em *New Project*, situado na seção *My Projects*, onde você também pode gerenciar seus projetos. É importante ressaltar que a ferramenta ainda é nova e está em desenvolvimento, ou seja, trata-se de uma versão Beta.

A plataforma oferece duas opções de linguagem para que você possa criar seus programas. Uma delas é a tradicional linguagem de blocos, com que já estamos familiarizados. Há uma diferença quanto à disposição, formato e estrutura dos blocos em relação à linguagem nativa, mas é mais uma questão estética e visual. A segunda linguagem oferecida é JavaScript. Se você quer ter contato com uma linguagem de programação textual, este é um bom começo.

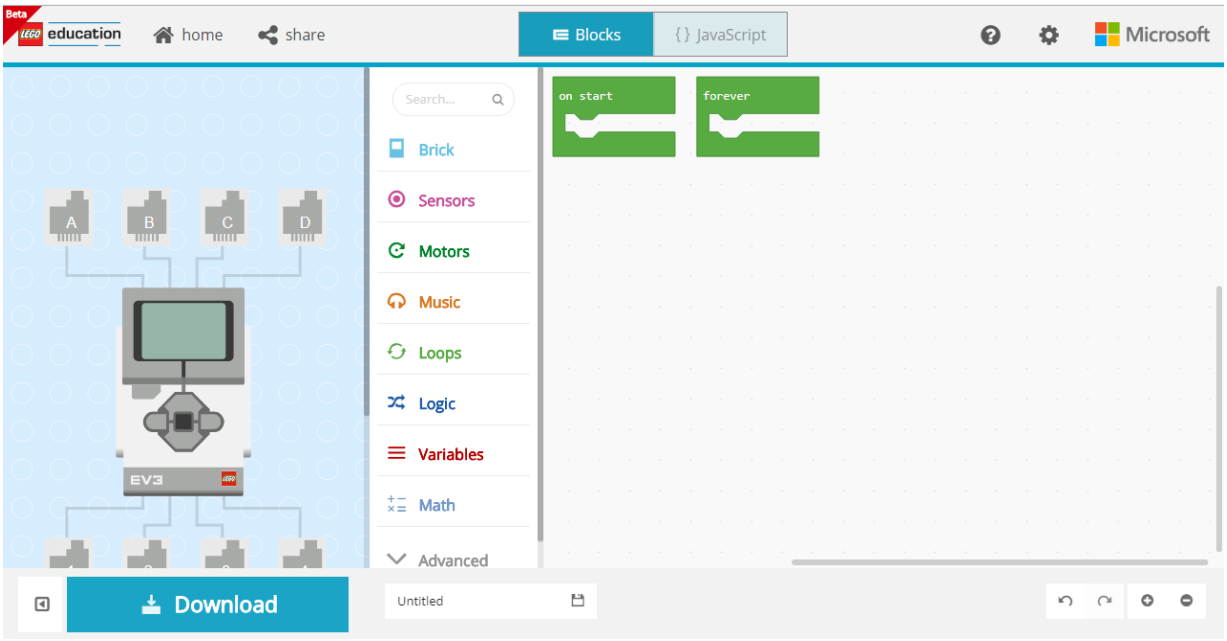



Figura 35.2: Microsoft Make Code: LEGO® MINDSTORMS® EV3 - Blocos

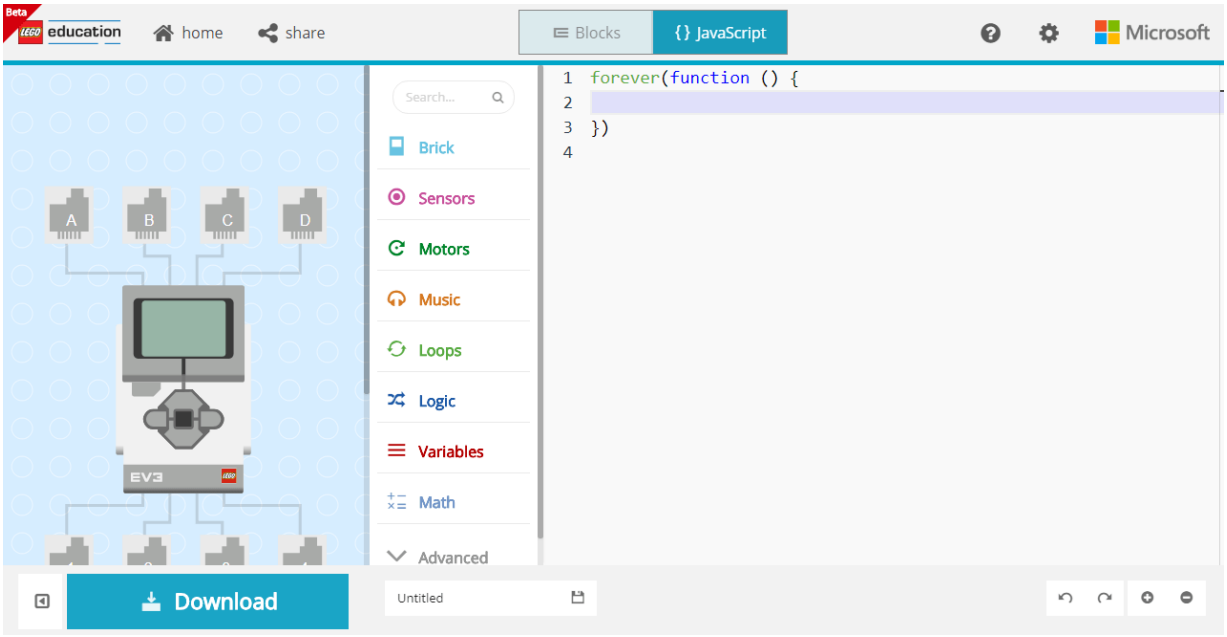



Figura 35.3: Microsoft Make Code: LEGO® MINDSTORMS® EV3 - JavaScript

# 35.2 ROBOTC

http://www.robotc.net/

ROBOTC é uma linguagem de programação baseada em C, desenvolvida para kits robóticos educacionais e competições de robótica. A linguagem possui uma IDE (ambiente de desenvolvimento integrado) própria. Você pode escolher entre programar utilizando a linguagem na sua forma gráfica (de blocos) ou textual. Entretanto, trata-se de um software pago.

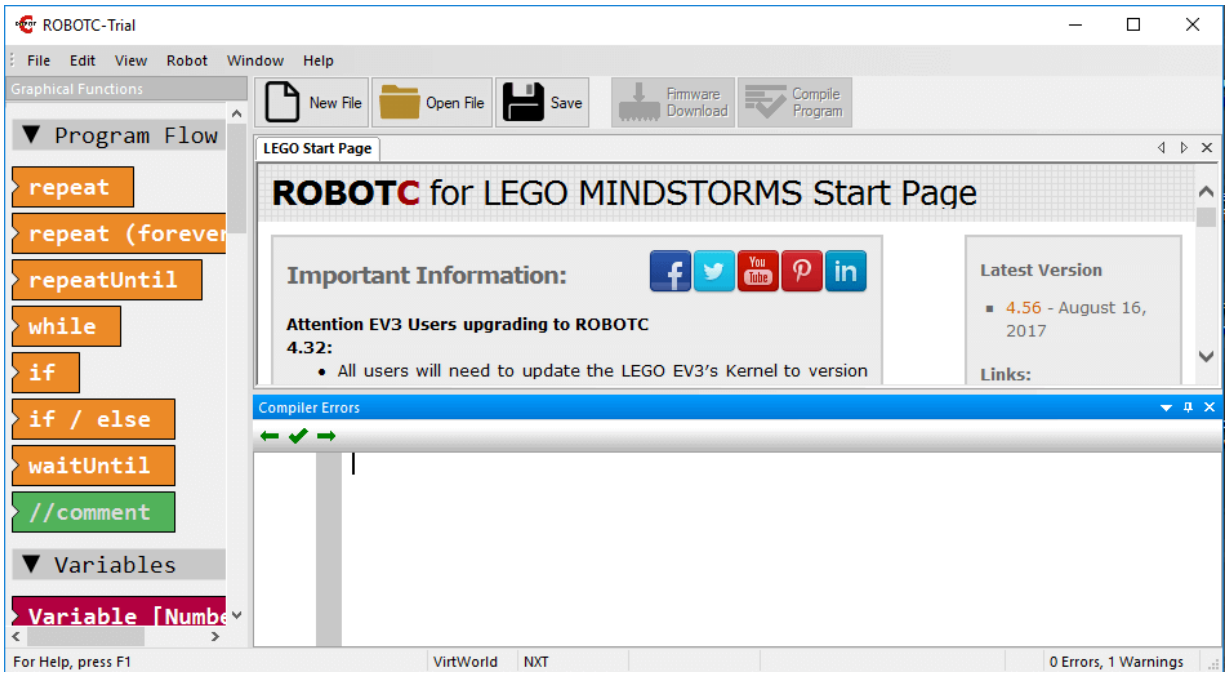



Figura 35.4: ROBOTC - Ambiente de desenvolvimento

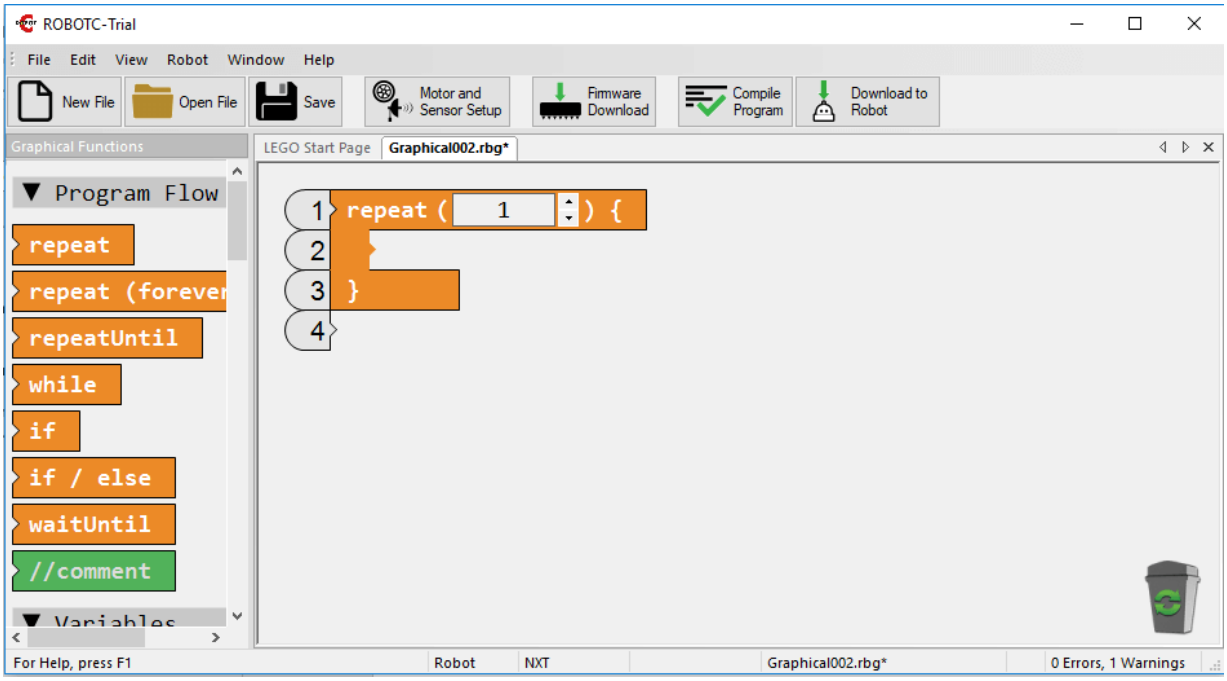



Figura 35.5: ROBOTC versão blocos - Ambiente de desenvolvimento

Um dos recursos mais atrativos no ROBOTC é que ele possui *Virtual Worlds*. Com eles, os Mundos Virtuais, você pode executar seus programas selecionando um dos vários robôs virtuais e um ambiente para seu robô como uma pista contendo uma linha negra, por exemplo.

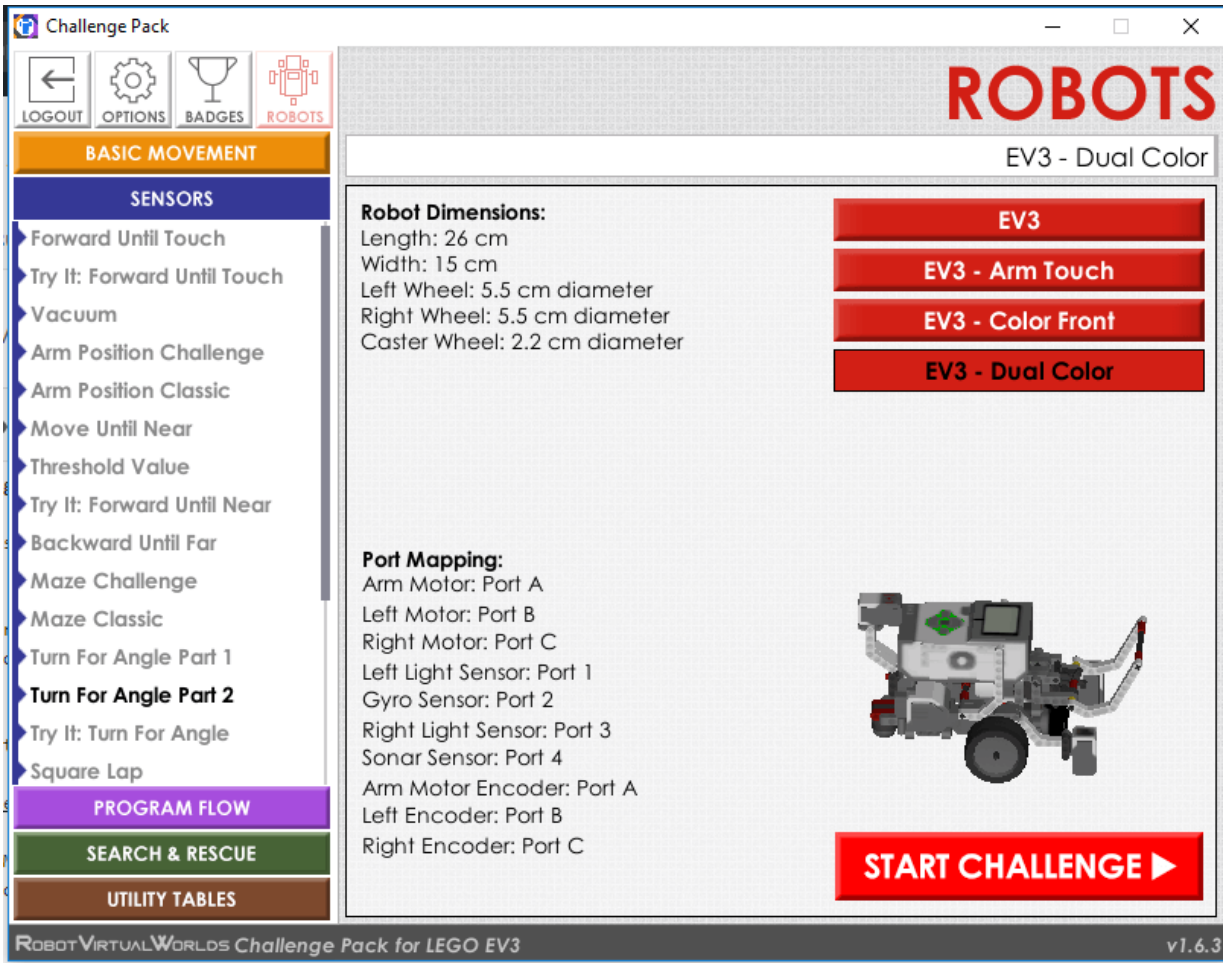



Figura 35.6: Virtual Worlds - Robots

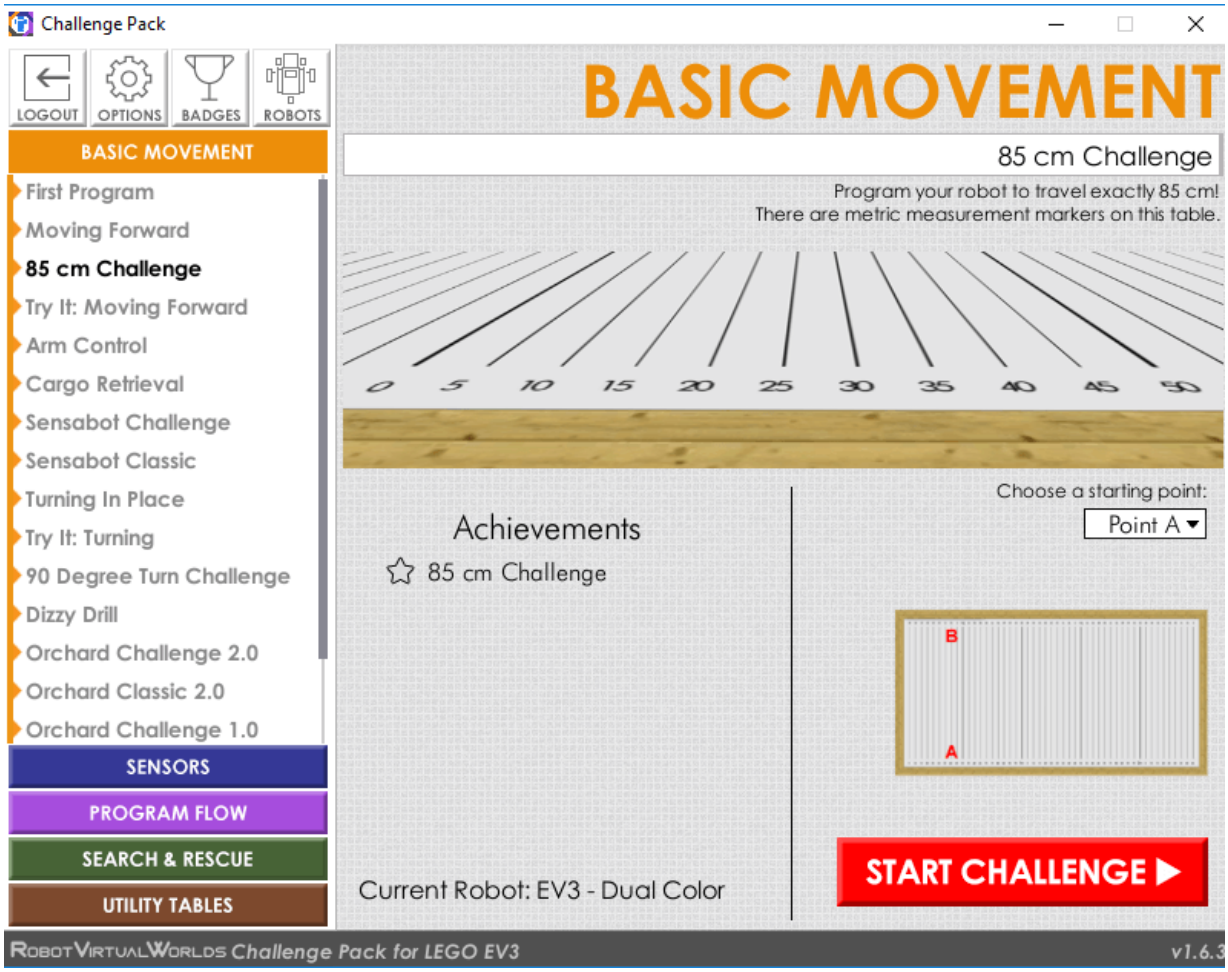



Figura 35.7: Virtual Worlds - Badges

Fazendo o comparativo entre o preço de uma licença do software e o custo de um kit LEGO® MINDSTORMS®, esta é uma ótima ferramenta para ensinar robótica e programação, principalmente se você pretende ministrar cursos ou aulas de robótica. Dessa forma, seus alunos podem programar e testar os códigos no próprio computador e, depois, com um kit você pode fazer a demonstração física.

### Virtual Brick

http://www.robotvirtualworlds.com/virtualbrick/

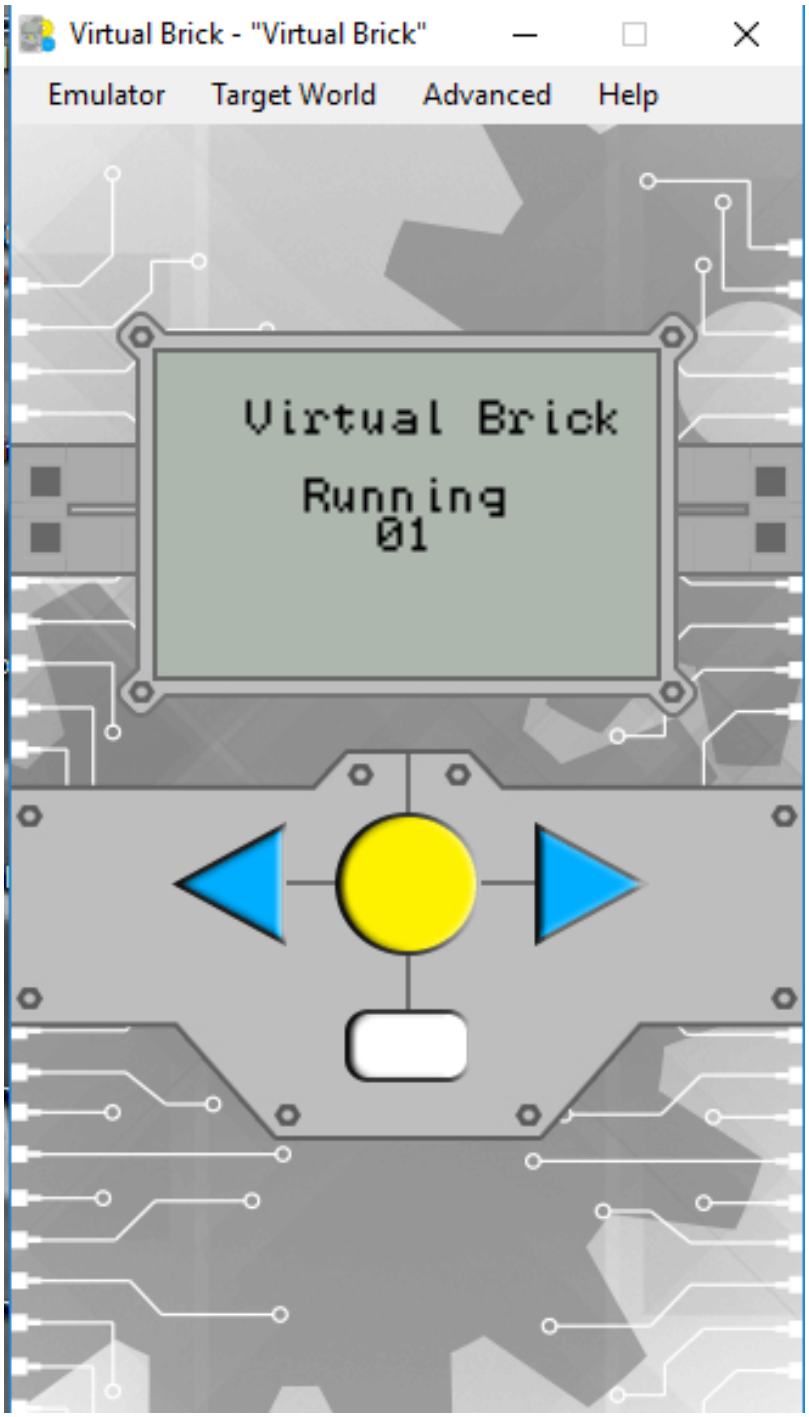



Figura 35.8: Virtual Brick

O Virtual Brick é um recurso incrível. Como o próprio nome sugere, ele simula um Brick, ou seja, é um Brick virtual. Ele age em ação conjunta com os mundos virtuais. A partir dele, é possível utilizar o

próprio software da para programar os robôs contidos nos Virtual Worlds e realizar as missões.

Com o software iniciado, basta realizar o download do programa para o Virtual Brick.

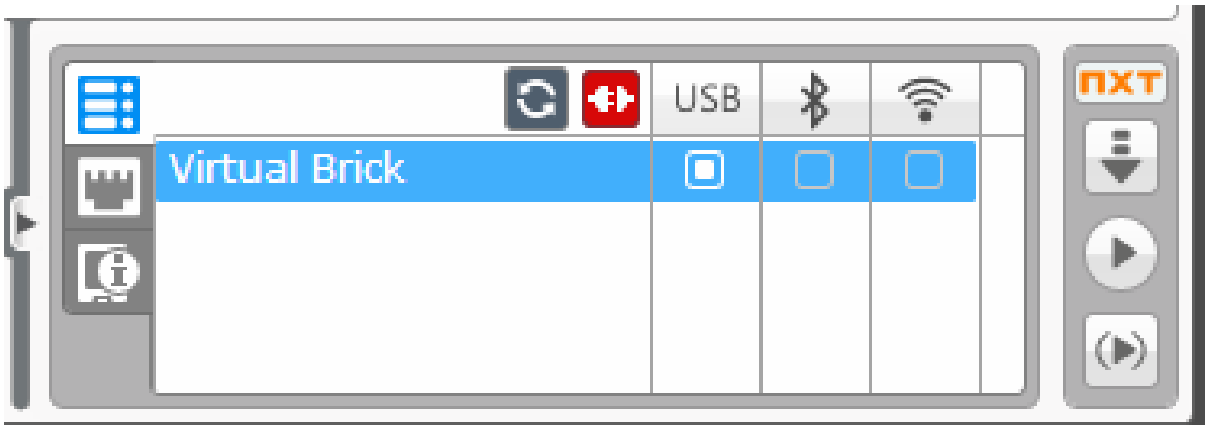



Figura 35.9: Virtual Brick

## 35.3 ev3dev

http://www.ev3dev.org/

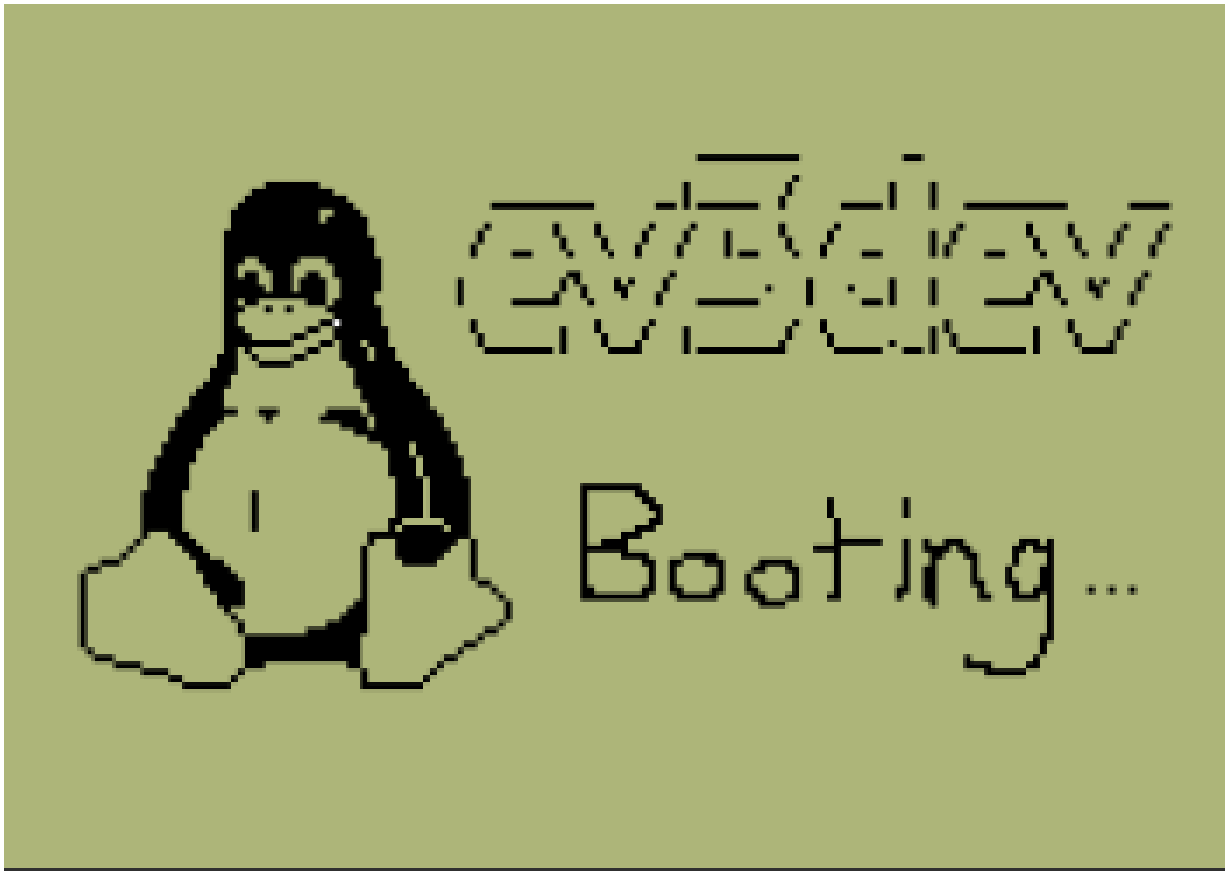



Figura 35.10: Virtual Worlds - Badges

O ev3dev é um sistema operacional baseado em Debian-Linux que roda no LEGO® MINDSTORMS® e outras plataformas compatíveis. Ele não é firmware, sua execução se dá através de um cartão micro SD inserido no *slot* do EV3. Sua instalação e preparação são um pouco mais trabalhosas, porém, como são feitas a partir do cartão SD, não é necessária nenhuma modificação direta no seu Brick EV3.

A principal vantagem do ev3dev é que ele é um sistema com suporte para diversas linguagens de programação. Além disso, por ser um sistema baseado em Linux, é totalmente livre e conta com uma comunidade muito participativa e colaborativa. São diversas bibliotecas que podem ser utilizadas e vários repositórios no GitHub contendo informações relacionadas à linguagem escolhida.

Dentre as linguagens, podemos citar: C, Java, Python, GO, JavaScript. Para ter acesso à lista completa de linguagens suportadas veja: http://www.ev3dev.org/docs/programming-languages/

Essas são ferramentas que ampliarão ainda mais seus conhecimentos tanto em robótica como programação. Aliás, a robótica é uma ótima forma de se aprender programação. Talvez, o melhor exemplo de quão vastas são as possibilidades seja o ev3dev, pois fornece um suporte a várias linguagens.

# Capítulo 36
# Nota final

Caro leitor,

Esta obra teve como intuito ser uma porta de entrada, um pontapé inicial para quem deseja aprender mais sobre robótica e sobre programação. O livro foi criado com o objetivo de ser um guia, por isso, logo no início, foram abordadas as diferenças entre os kits, os seus softwares e as ferramentas presentes no software. Compreender a ferramenta de trabalho e suas possibilidades é fundamental para uma boa aprendizagem.

Antes de iniciar a programação, você aprendeu tudo o que foi necessário para melhorar sua curva de aprendizagem. Por meio de programas envolvendo apenas o Brick, até robôs mais sofisticados, você aprendeu a utilizar diversos blocos, constantes, variáveis e muito mais. Além de aprender robótica de uma maneira bem divertida, você também aprendeu lógica de programação, que é o ponto de partida para qualquer linguagem de programação, seja ela de blocos ou não.

O intuito deste livro foi iniciar você, leitor, em uma caminhada, ajudá-lo em seus primeiros passos. O livro chegou ao fim, mas sua jornada não. Intencionalmente, antes deste capítulo final, apresentamos uma espécie de roteiro de cunho sugestivo. O motivo para tal foi o desejo de que você continue seus estudos, com o alcance das mais vastas possibilidades do seu kit. São sugestões do que ainda pode ser feito por você, quais podem ser seus próximos passos.

Espero que você esteja com um turbilhão de ideias, prontas (ou quase) para serem implementadas. Que você ainda esteja sedento e faminto por desvendar cada bloco, cada peça, cada componente do seu kit, e sempre pronto para superar novos desafios.

Obrigado a todos!

- Vitor Emanuel Gonçalves e Luiz Alberto Roque

# CAPÍTULO 37
# Links úteis

## 37.1 Manuais e instruções de montagem

### Home Edition

Guia do usuário:

https://www.lego.com/en-us/mindstorms/downloads/user-guide

Instruções de montagem:

https://www.lego.com/pt-br/service/buildinginstructions/search?initialsearch=31313#?text=31313

### Education

Guia do usuário:

https://education.lego.com/en-gb/support/mindstorms-ev3/user-guides

Instruções de montagem:

https://education.lego.com/en-gb/support/mindstorms-ev3/building-instructions

### Informações sobre o software

Para informações em geral, informações sobre as ferramentas e descrição detalhada de cada bloco, seu funcionamento e configuração, acesse:

**LEGO® MINDSTORMS® EV3 Help**

https://ev3-help-online.api.education.lego.com/Retail/en-us/page.html?Path=editor%2Findex.html